MATLAB 与化学

——作图、计算与数据处理

郑冀鲁　编著



·北京·

本书讲述了如何利用MATLAB解决化学工作者遇到的作图、计算与数据处理问题。内容包括MATLAB简介与简单计算、MATLAB作图、非线性方程求解、线性方程组求解、常微分方程与方程组的求解、插值、数值积分与数值微分、最优化、非线性方程组求解、曲线拟合、统计分析十一个专题。本书一方面讲述了解决上述问题的基本原理，使得读者做到知其然亦知其所以然，另一方面讲述如何使用MATLAB中的各种函数来解决上述问题而不需要读者自行编写程序代码。此外在每个专题中均附带若干实例，实例中的问题长期困扰着广大化学工作者，而本书给出了这些问题较为圆满的解决方案。

本书可作为化学专业教学、科研人员进行作图、化学计算和数据处理的参考书，也可作为化学及相关专业学生的参考材料。

图书在版编目（CIP）数据

MATLAB与化学——作图、计算与数据处理/郑冀鲁编著．北京：化学工业出版社，2009.2
ISBN 978-7-122-04132-6

Ⅰ．M…　Ⅱ．郑…　Ⅲ．化学-计算机辅助计算-软件包，MATLAB　Ⅳ．06-39

中国版本图书馆CIP数据核字（2008）第178523号

责任编辑：傅聪智　　装帧设计：张　辉
责任校对：徐贞珍

出版发行：化学工业出版社（北京市东城区青年湖南街13号　邮政编码100011）
印　　装：三河市延风印装厂
720mm×1000mm　1/16　印张 $11\frac{3}{4}$　字数231千字　2009年1月北京第1版第1次印刷

购书咨询：010-64518888（传真：010-64519686）　售后服务：010-64518899
网　　址：http://www.cip.com.cn
凡购买本书，如有缺损质量问题，本社销售中心负责调换。

定　　价：**36.00**元

前　言

化学工作者在科研与教学过程中常常会遇到许多作图、科学计算和数据处理问题，这些问题遍及化学的各个分支，如做出某种多元酸的各种存在形式的分布曲线，根据实验数据做出折线图，以反映了一个变量随另外一个变量的变化趋势或关系，计算反应的焓变、计算复杂反应体系的化学平衡、计算酸碱溶液的 pH 值、间歇反应器中反应物浓度随时间的变化，根据离散的反应物浓度-时间关系数据求解任意时刻的反应速率等。这些问题从数学角度可以归结为函数作图（离散的或连续的），数学四则运算、非线性方程与方程组的求解、插值、数值积分与数值微分、求解微分方程等科学计算问题。简单的作图、科学计算和数据处理一般以使用 Excel 完成（学习 Excel 在这方面的功能也需要花费不少时间），但其它较为复杂的函数作图、科学计算和数据处理问题如求解非线性方程与方程组等则需要使用 Fortran，C，Basic 等高级语言编制计算机程序来解决。但化学工作者由于专业关系，没有系统学习和实践过利用计算机高级语言编制程序解决科学计算问题的课程，数学基础也比较薄弱，因此造成广大化学工作者无法有效解决在各化学分支遇到的各种复杂的作图、科学计算和数据处理。

MATLAB 的出现使得广大化学工作者能够轻易地解决在各化学分支遇到的各种复杂的作图、科学计算和数据处理问题。

MATLAB 是美国 Mathworks 公司开发的科学计算与作图软件。MATLAB 包含了大量的函数，通过调用一个或几个函数再加上一些流程控制语句，化学工作者就能够解决各种各样的复杂数学计算和作图问题，而在以前，求解这些复杂的数学计算和作图问题需要编写包含数十行甚至数百行代码的高级语言程序。

就科学计算而言，依作者看来，MATLAB 不仅是一种方便的科学计算软件，更是一种优秀的计算机语言。作为一种高级计算机语言，MATLAB 与其它语言显著不同的特点在于：①MATLAB 能够直接对矩阵操作；②MATLAB 中能够直接使用复数。瑞典数学家 Lars Garding 在其名著“Encounter with Mathematics”中说：“如果不熟悉线性代数的概念，要去学习自然科学，现在看来就和文盲差不多。”确实，线性代数不仅在数学理论和各个自然科学领域中起着巨大的作用，而且在科学计算中占有重要地位，许多科学计算问题最后往往转化为线性代数问题。矩阵是线性代数的核心概念，线性代数的问题其实就是矩阵分析问题。因此不难理解，MATLAB 能够直接对矩阵操作，将会为解决科学计算问题带来多么巨大的便利。作者相信读者在使用 MATLAB 的过程中，将会越来越强烈地感受到 MATLAB 能够直接对矩阵操作这一特点具有多么大的优越性，利用这一特点，编写程

序是多么方便与自然。虽然大部分数学分支仅仅使用实数，但复数才是数学发展的完善阶段，复数体现了数学的统一与完备之美。但目前几乎所有的高级语言都不支持复数，对负实数开偶次方根或对负实数取对数被很多高级语言认为是非法的，而在 MATLAB 中，对负实数开偶次方根或对负实数取对数被认为是合理的，MATLAB 将给出正确的运算结果，结果自然是复数。由此可见，MATLAB 确实是为科学计算而诞生的语言，它充分考虑到了科学计算的需要，处处为从事科学计算的科研人员提供方便。

除了科学计算问题外，化学工作者还会遇到数据处理问题和作图问题。如将若干数据点拟合为直线，计算样本的均值和标准差，求置信度与平均值的置信区间，对两组样本的某些数字特征进行比较并进行假设检验；画出某个函数的图形，将离散数据点根据需要表达为折线图、柱形图、饼图或直方图。MATLAB 提供了丰富的数据处理和作图函数以解决化学工作者遇到的上述各种问题。

相信读者在使用 MATLAB 的过程中，一定会体会到它不同于其它计算软件或高级计算机语言的优越性，会逐渐喜爱上它。

本书以在化学科研与教学遇到的各种科学计算、作图、数据处理问题为核心，将这些问题分成若干专题来叙述。因为只有这样，才能抓住遍布于化学中出现的大量计算、作图、数据处理问题的本质，起到纲举目张的效果。因此，本书在形式上由 MATLAB 简介与简单计算、MATLAB 作图、非线性方程求解、线性方程组求解、常微分方程与方程组的求解、插值、数值积分与数值微分、最优化、非线性方程组求解、曲线拟合、数据分析十一个专题构成，在内容上一方面讲述解决上述问题的基本原理，使得读者做到知其然亦知其所以然，另一方面讲述如何使用 MATLAB 中的各种函数来解决上述问题而不需要读者自行编写程序代码。此外在每个专题中均附带若干实例，如根据离散的反应物浓度-时间关系数据求解任意时刻的反应速率、利用已有的实验数据估计出未知的实验数据、复杂反应体系的化学动力学计算、复杂体系化学平衡的计算、溶液的 pH 值的计算和确定电位滴定终点等。这些问题长期困扰着广大化学工作者，而本书给出了这些问题较为圆满的解决方案。从创新性角度看，本书提出了采用山丘样条插值算法进行数值积分和微分，而数值积分和数值微分在化学实验数据处理方面具有特别重要的用处。本书还提出了采用遗传算法＋局部优化方法求解最优化问题，这就为求解非线性方程组和非线性最小二乘法奠定了坚实基础，复杂反应体系的平衡计算实际就是求解非线性方程组，将实验数据拟合为非线性方程则要使用非线性最小二乘法。

作者在使用 MATLAB 的过程中，根据实践的需要，也编制许多 MATLAB 函数来解决科学计算问题，作者认为在某些科学计算问题中，作者自己根据实际编写的函数能更有效、更方便地解决这些科学计算问题。因此在这些专题中，作者给出了自编的函数及程序代码，供读者在实践中使用。虽然使用 MATLAB 和作者提供的各种函数，能够基本解决读者在化学实践中遇到的各种化学计算、作图与数据分

析问题而不需要读者亲自去编程，但作者仍然竭力建议读者学习利用 MATLAB 编制程序来解决各种问题的知识与方法。因为现实问题千变万化，如果仅仅会使用一些固有函数，那么势必造成读者在面对新的问题时束手无策。只有学会编制 MATLAB 程序，利用流程控制语句和其它语句将 MATLAB 的固有函数和自编函数组合起来，才能充分发挥 MATLAB 的能力，从而极大地拓展读者使用 MATLAB 解决在化学实践中遇到的各种问题的能力。

欢迎广大读者来函，与作者就 MATLAB 在化学中的应用进行广泛、深入的交流！作者 E-mail：triace@163.com。

由于作者水平有限，本书必然存在种种不足之处，恳请读者批评指正！

郑冀鲁

2008 年 10 月于郑州大学

目　录

p



MATLAB是Mathworks公司开发的专门面向科学计算的软件和语言。目前MATLAB的最新版本是MATLABR2008a，国内较为流行的版本是MATLAB6.5和MATLAB7.0。MATLAB6.5和MATLAB7.0均是功能较完善的版本，但MATLAB7.0具有更强和更完善的科学计算功能。因此，本书介绍的各种函数和代码以MATLAB7.0为基础，它们均能在MATLAB7.0环境下顺利运行。

MATLAB科学计算、作图、数据处理功能非常全面，内容丰富而庞杂。如果读者一开始就去全面而详细的了解MATLAB的各种功能，会使得读者只见树木、不见森林，迷失在庞杂的MATLAB函数里，并不能使读者熟练掌握MATLAB的使用方法和技巧。因此本章将介绍MATLAB中最基本、最重要的内容，读者掌握了这些最基本的东西后，再结合后面的各章节所讲的内容，就能够使用MATLAB解决日常化学实践中遇到的各种科学计算、作图和数据处理问题。如果遇到新的问题，也能以本书所讲的内容为基础，通过查阅其它MATLAB的资料或通过MATLAB的在线帮助，独立地解决遇到的新问题。

1.1 MATLAB的安装

由于现在计算机极其普及，读者对应用软件的安装都比较熟悉，因此，本部分仅提及安装MATLAB时读者最需要注意的部分。

MATLAB由MATLAB本体和数十个MATLAB工具箱组成。MATLAB本体包括各种语句、命令、运算符和基本函数，可以毫不夸张地说，仅仅使用MATLAB本体，就能完成绝大部分作图、科学计算和数据处理任务。每个MATLAB工具箱是一组具有特定功能的MATLAB函数的集合，均能完成某个特定领域的任务，如MATLAB的统计学工具箱包含一组专门处理数理统计问题的函数。MATLAB工具箱是MATLAB功能的扩展，它们需要在MATLAB本体的支持下发挥自身的作用。

对于MATLAB7.0而言，如果把MATLAB本体和数十个工具箱全部安装，需要2GB以上的硬盘空间。自然，MATLAB本体是非装不可的，但我们不必将MATLAB的几十个工具箱全部安装，事实上，我们只需要安装最常用的MATLAB工具箱，一般包括统计学工具箱（Statistics Toolbox）、最优化工具箱（Optimization Toolbox）、偏微分方程工具箱（Partial Differential Equation Toolbox）、

曲线拟合工具箱（Curve Fitting Toolbox）、遗传算法工具箱（Genetic Algorithm Direct Search Toolbox）、样条函数工具箱（Spline Toolbox）。

选择安装这些工具箱的步骤如下。

在执行MATLAB安装程序，输入安装序列号后，按照安装程序的提示依次执行，将会出现“安装类型”窗口，见图1-1，在此处，应该用鼠标点击“Custom”前面的单选框（○），进行自定义安装。

用鼠标点击“Custom”前面的单选框后，点击next，就进入了“产品和文件夹”选择窗口，见图1-2，在这个窗口，可以指定将MATLAB安装在哪个目录下，

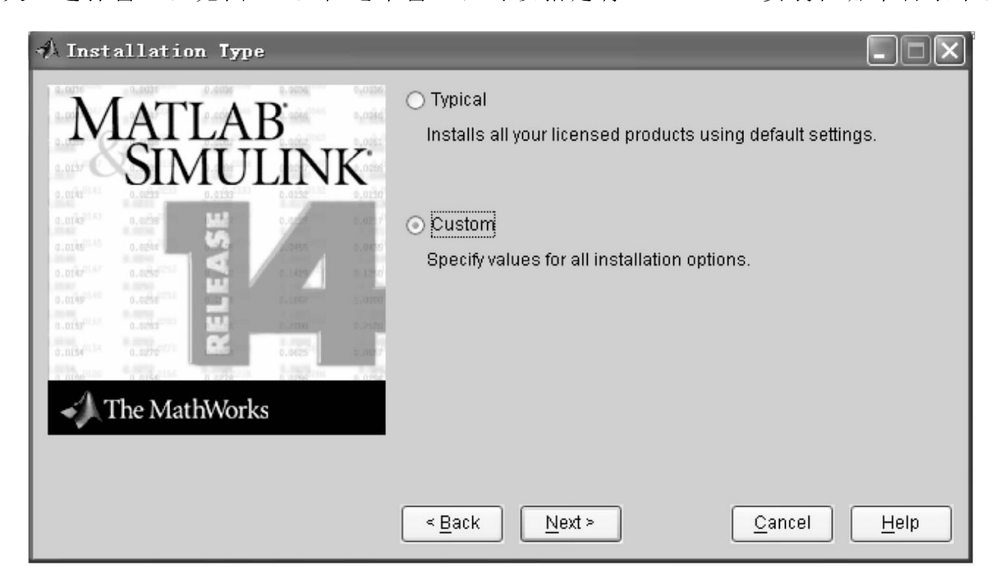


图1-1 “安装类型”窗口

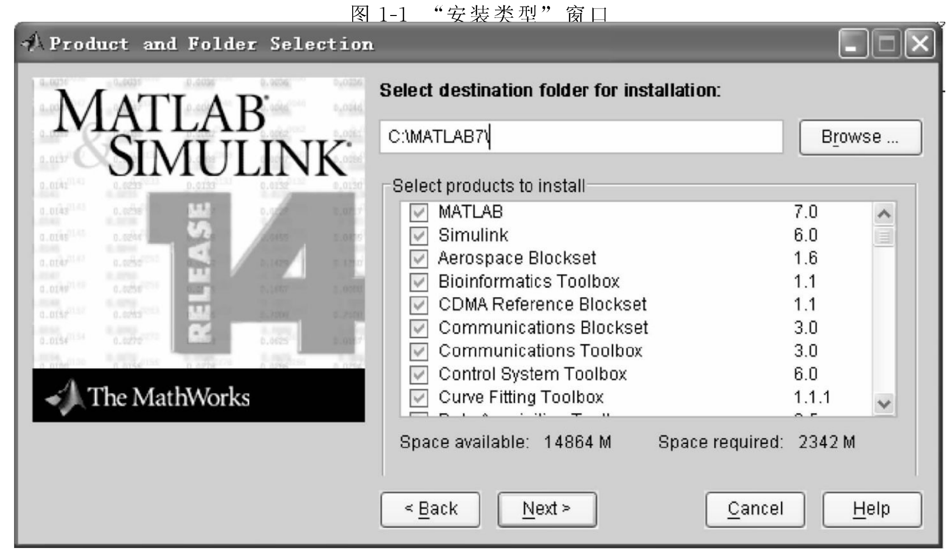


图1-2 “产品和文件夹”选择窗口

安装哪些工具箱。在该窗口的最上方的长矩形框内，可以输入 MATLAB 的安装文件夹，也可以点击旁边的“Browse”按钮，选择希望的 MATLAB 安装文件夹。在该窗口下方的大矩形框内，可以选择需要安装的工具箱，方法是用鼠标点击每个产品（包括 MATLAB 本体和数十个 MATLAB 工具箱）前面的复选框（□），在复选框（□）里打上对勾（√），如此，该复选框（□）对应的产品将被选择，会安装到硬盘，如果不希望某工具箱被安装，而此工具对应的复选框里已经打上对勾，这时只需用鼠标点击该复选框，则该复选框里的对勾将被去掉，该工具箱将不会被安装。

1.2　一个简单的计算实例

运行 MATLAB 后，将看到 MATLAB 的界面，见图 1-3。

在这个界面里，最重要的是右面的“Command Window”窗口，在右面的“Command Window”窗口可以输入各种函数和语句，进行化学计算、作图和数据处理。

假如我们需要计算：欲配制 0.02000mol/L 的 $K_2Cr_2O_7$ 标准溶液 250mL，应称取 $K_2Cr_2O_7$ 多少克？

为了完成计算，需要在 MATLAB 的“Command Window”窗口依次输入图 1-4 所示语句。

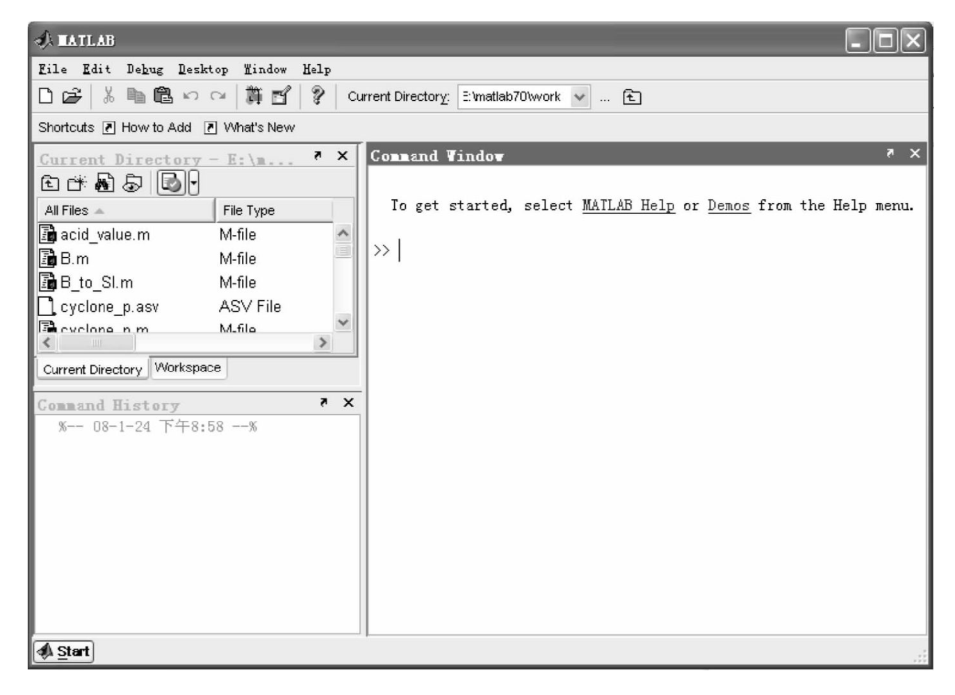


图 1-3　MATLAB 的运行界面

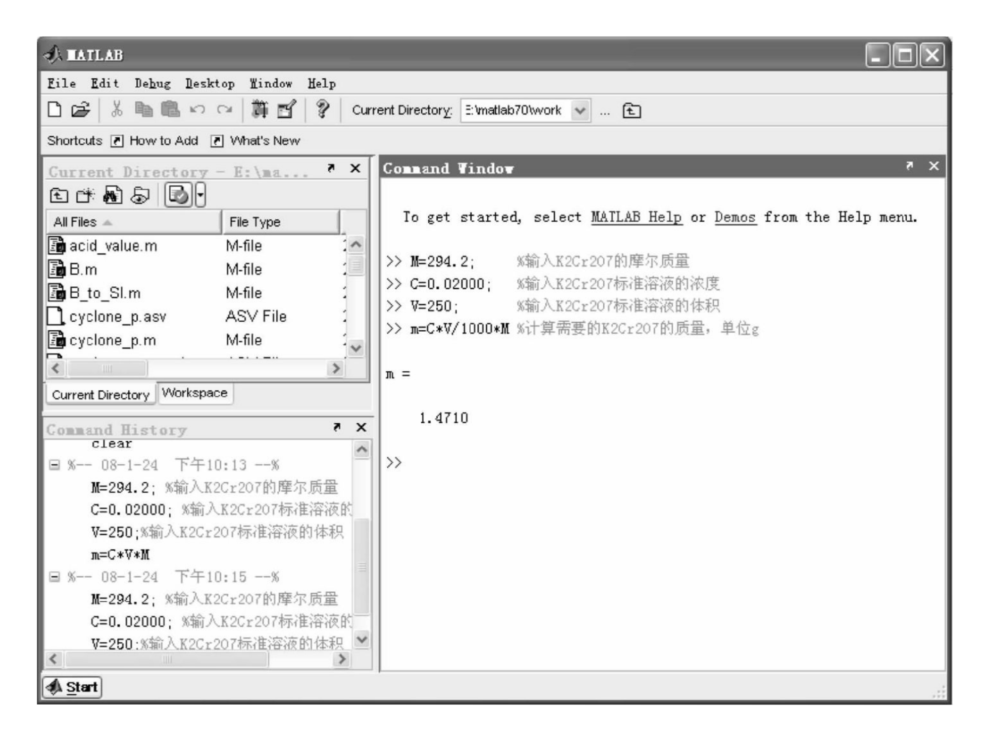


图 1-4　MATLAB 的计算结果

每输入一行，敲回车，然后输入下一行。依次输入完上述语句，最后敲回车，MATLAB 将给出最后计算结果，见图 1-4。

双大于号（≫）是 MATLAB 的提示符，当“Command Window”窗口出现双大于号（≫）时，表示 MATLAB 处于等待状态，准备执行计算任务。M、C、V 和 m 既是变量，又是变量名，代表实际问题中某个物理量，它们能够被赋值，赋值后可以参加各种运算。MATLAB 的变量名由字母、数字和下划线组成，由字母打头。如 *m1*、*ab*、*A_1*、*acid_value* 等都是合法变量名。等号（=）是赋值号，其作用是将一定数值赋给变量。星号（*）在 MATLAB 中代表乘号，斜杠（/）在 MATLAB 中代表除号。请注意语句“M=294.2;”和“m=C*V/1000*M”的区别，这两个语句在本质上是一样的，作用是把一定的数值赋给指定变量。其区别就是前一个语句后面有分号（;）后面一个语句没有分号，对于加有分号的赋值语句，MATLAB 仅仅将数值赋给指定变量而不执行其它操作，对于没有分号的赋值语句，MATLAB 不仅执行赋值操作，而且还会输出被赋值变量的数值。% 是 MATLAB 的注释符号，在注释符号后面可以输入各种说明性文本。% 连同说明性文字通常用在 MATLAB 的一条语句后面或某一函数的开头，说明语句或函数的作用。因为人总有遗忘性，对于自己编的程序或函数，在过了较长一段时间后就会忘记函数或程序的作用、如何使用和为什么这么编。因此使用注释符号加说明文本是个良好的习惯。

1.3　矩阵与向量的定义

MATLAB 的特点之一就是能够直接操作矩阵，这是 MATLAB 在科学计算方面具有极大优势的原因之一。矩阵是一种二维数据结构，具有行和列的维度。向量（包括行向量和列向量）以及数，可以看做矩阵的特殊情形或退化情形。

我们首先需要知道的是，在 MATLAB 中如何定义一个矩阵

考虑一个 3 行 4 列的矩阵 $a=\begin{matrix}1 & 2 & 6 & 4\\ 5 & 9 & 7 & 8\\ 9 & 4 & 1 & 2\end{matrix}$

MATLAB 中，用分号“;”来分开矩阵的各行，用空格或逗号（,）来区分矩阵的各列，不管是任何矩阵，我们可以直接按行方式输入每个元素：同一行中的元素用逗号（,）或者用空格符来分隔，且空格个数不限；不同的行用分号（;）分隔。所有元素处于一方括号（ [ ]）内。因此 3 行 4 列的矩阵 a 在 MATLAB 中定义：

a=[1 2 6 4; 5 9 7 8; 9 4 1 2] 或 a=[1, 2, 6, 4; 5, 9, 7, 8; 9, 4, 1, 2]

在 MATLAB 中定义矩阵 a 及运行结果如下：

```
≫a=[1 2 6 4;5 9 7 8;9 4 1 2]
   1   2   6   4
a=5   9   7   8
   9   4   1   2
```

或：

```
≫a=[1,2,6,4;5,9,7,8;9,4,1,2]
   1   2   6   4
a=5   9   7   8
   9   4   1   2
```

向量是只有一行或只有一列的矩阵，因此向量在 MATLAB 中的定义完全遵照矩阵定义的规则来定义。如行向量 $a=(1\ 3\ 9\ 5)$ 在 MATLAB 中定义为 a=[1 3 9 5] 或 a=[1, 3, 9, 5]。

列向量 $b=\begin{matrix}3\\ 7\\ 1\\ 9\end{matrix}$ 在 MATLAB 中定义为 b=[3; 7; 1; 9]。

在 MATLAB 中定义行向量 a 和列向量 b 及运行结果如下：

```
≫a=[1 3 9 5]
a=1   3   9   5
≫b=[3; 7; 1;9]
```

```
    3
b = 7
    1
    9
```

接下来我们要谈一谈如何快速定义一类特殊的向量，这些向量的各个分量依次构成了一个等差数列。如行向量 p（1 3 5 7 9 11）如何快速定义？这个行向量的各分量依次构成了首项为1，公差为2，末项为11的等差数列。MATLAB采用冒号算符（:）来定义这类特殊行向量。在MATLAB环境输入如下语句：

≫p＝1:2:11

敲回车，运行该语句，得到

p＝1　3　5　7　9　11

因此，我们看出，在MATLAB中快速定义这类行向量，要知道向量的第一个分量，分量公差，向量的最后一个分量，然后通过冒号算符（:），将要定义的向量写作:

向量的第一个分量:分量的公差:向量的最后一个分量

如果分量的公差为1，在冒号算符（:）连接的向量书写格式中，公差1可以省略，那么要定义的向量写作：

向量的第一个分量:向量的最后一个分量

如在MATLAB中输入语句：

≫p＝2:6

回车，得到

p＝2　3　4　5　6

那么对于各分量构成了一个等差数列的列向量，该如何定义呢？可以先用冒号算符（:）获得一个各分量构成一个等差数列的行向量，然后利用MATLAB的矩阵转置算符（'）将行向量转换为列向量。

如打算定义如下列向量：

$$b=\begin{matrix}1\\4\\7\\10\end{matrix}$$

可在MATLAB中输入如下语句：

≫b＝1:3:10；

≫b＝b'

敲回车，得：

```
    1
b = 4
    7
   10
```

从另外一个角度看，形如 a:d:b 格式的行向量实际是由下面分量依次构成的行向量：

a、a＋d、a＋2d、…、a＋(n－1)d、a＋nd、…

当 d＞0 时，如果 a＋nd＞b，a:d:b 格式的行向量实际是［a a＋d a＋2d…a＋(n－1)d］，如果 a＋nd＝b，a:d:b 格式的行向量才是（a a＋d a＋2d…b）。同样，当 d＜0 时，如果 a＋nd＜b，a:d:b 格式的行向量实际是［a a＋d a＋2d…a＋(n－1)d］，如果 a＋nd＝b，a:d:b 格式的行向量才是（a a＋d a＋2d…b）。

考察语句：

≫m＝1:2:10

敲回车，得到：

m＝1　3　5　7　9

考察语句：

≫n＝10:－2:1

敲回车，得到：

n＝10　8　6　4　2

我们来看一种有趣的情形，比如在 MATLAB 中输入：

≫p＝2:3:1

向量 p 会等于什么。根据我们上面提到的规则，甚至向量 p 的第一个分量 2 也不应该包括在向量 p 中，似乎向量 p 不应该包含任何一个分量。这样的向量存在吗？存在，在 MATLAB 中这种特殊不包含任何分量的向量称为空向量。那么上述的向量 p 是空向量吗？我们不妨亲自在 MATLAB 中验证一下。输入上述语句后，敲回车，得到

p＝Empty matrix: 1-by-0

MATLAB 显示 p 是一个 1 行 0 列的空矩阵，实际就是个空行向量。

同样，对于语句：

≫q＝5:－2:9

敲回车，得到：

q＝Empty matrix:1-by-0

q 也是一个 1 行 0 列的空矩阵，实际上也就是空行向量。

因此，对于形如 a:d:b 格式的行向量，当 d＞0 时，如果 a＞b，那么该向量实际是个空向量。当 d＜0 时，如果 a＜b，那么该向量实际也是个空向量。

下面介绍几种特殊矩阵的生成方法。

空矩阵：不包含任何元素的矩阵

空矩阵在 MATLAB 中用［ ］表示，如下的语句可以让变量 a 代表空矩阵：

≫a＝[ ]

敲回车，得到：

a= [ ]

零矩阵：所有元素都是整数 0 的矩阵。

零矩阵使用 MATLAB 的 zeros 函数生成，比如希望生成一个 2 行 3 列的零矩阵 b，则输入如下 MATLAB 语句：

```
≫b=zeros(2,3)
```

敲回车，得到：

```
b=0  0  0
  0  0  0
```

全 1 矩阵：所有元素都是 1 的矩阵。

全 1 矩阵使用 MATLAB 的 ones 函数生成，比如希望生成一个 2 行 3 列的全 1 矩阵 b，则输入如下 MATLAB 语句：

```
≫b=ones(2,3)
```

敲回车，结果如下

```
b=1  1  1
  1  1  1
```

单位矩阵：对角线元素是 1，其余元素是 0 的矩阵。

单位矩阵都是方阵。MATLAB 中生成单位矩阵的函数是 eye，如希望生成一个 3 行 3 列的单位矩阵 b，则输入如下 MATLAB 语句：

```
≫b=eye(3)
```

敲回车，得到

```
  1  0  0
b=0  1  0
  0  0  1
```

1.4 矩阵合并

MATLAB 允许将若干小矩阵合并为一个更大的矩阵，这些小矩阵从表观上就像普通的矩阵元素一样，通过空格或逗号（,）来区分列，通过分号（;）来区分行，这些小矩阵通过空格或逗号和分号连接为一个大矩阵。能够合并为一个大矩阵的各个小矩阵，它们的行数和列数必须符合一定的数量关系。

假设小矩阵 A，B，C，D 可以合并成一个大矩阵 M，如下：

$$M=\begin{matrix} A & B \\ C & D \end{matrix}$$

那么，矩阵 A 的行数必须和矩阵 B 的行数相等，矩阵 C 的行数必须和矩阵 D 的行数相等；矩阵 A 的列数必须和矩阵 C 的列数相等，矩阵 B 的列数必须和矩阵 D 的列数相等。

上述矩阵合并，写成 MATLAB 语句就是：

M=[A B；C D] 或 M=[A，B；C，D]

当然，既然矩阵 A 的行数必须和矩阵 B 的行数相等，A 和 B 也可以合并为一个大矩阵 [A B]，由于矩阵 A 的列数必须和矩阵 C 的列数相等，A 和 C 可以合并为大矩阵 [A；C]。

1.5　引用矩阵中的元素与矩阵块

在了解矩阵和向量如何定义、如何进行合并后，接着我们需要知道的是如何引用矩阵或向量中的某一个元素。

考虑矩阵 a=

11	32	54
61	28	10
34	56	21

如果想将矩阵 a 中的元素 10 赋给变量 b，该如何做呢？因为元素 10 在位于矩阵 a 的第 2 行，第 3 列，所以，写下如下的 MATLAB 的语句：

≫b=a(2,3)

敲回车，得到

b=10

对于一个行向量 p=[2 3 8 9 10]，如果想将向量 p 中的分量 8 赋值给变量 q，该怎么做呢？根据上面引用矩阵元素的例子，由于 8 是向量 p 的第 3 个分量，下面的语句当然是对的。

≫q=p(1,3)

敲回车，得到：

q=8

然而，对于向量来说，更简洁的引用方法是：

≫q=p(3)

敲回车，得到：

q=8

现在我们来学习矩阵块的引用与赋值。所谓矩阵块是一个矩阵的某个矩形部分，一个矩阵的某一行或某一列是矩阵块的特殊情形。我们看一下如何把一个矩阵的某个矩阵块整体提取出来然后赋值给某个变量。

考虑矩阵 a=

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

如果打算将矩阵 a 中用椭圆圈住的矩形块整体提取出来，赋值给另外一个变量 b，该如何做呢？通过观察，可以看出该矩形块对于该矩阵来说，在行的方向属于第3、4行，在列的方向属于第2、3、4列，因此可写下下面的MATLAB的语句：

```
≫b=a(3:4,2:4)
```

敲回车，得到：

```
   12   13   14
b=
   17   18   19
```

请注意，如下的语句：

```
≫b= a([3 4],[2 3 4])
```

敲回车，也可以得到相同的结果。

如果我们想把矩阵 a 的整个第2行赋值给变量 b，自然，经过赋值后，变量 b 就成为一个行向量。依据上面的例子，MATLAB语句如下：

```
≫b=a(2,1:5)
```

敲回车，得到：

```
b=6   7   8   9   10
```

因为是引用矩阵 a 的整行元素，因此下面的语句更简洁，更常用：

```
≫b=a(2,:)
```

敲回车，得到：

```
b=6   7   8   9   10
```

如果想整体提取矩阵 a 的第3列，然后赋值给变量 c，则MATLAB语句如下：

```
≫c=a(:,3)
```

敲回车，得到：

```
   3
   8
c=13
   18
   23
```

1.6 矩阵元素与矩阵块的赋值

如果想给矩阵里的某个元素或向量中的某个分量赋值，该如何去做呢？假设有一2行3列的零矩阵 c，想将零矩阵 c 的第1行第2列的元素重新赋值为1，输入如下MATLAB语句：

```
≫c=zeros(2,3);
≫c(1,2)=1
```

敲回车，得到

$$c=\begin{matrix}0 & 1 & 0\\ 0 & 0 & 0\end{matrix}$$

假设向量 b 是含有 6 个分量都是 1 的行向量，现在想把向量 b 的第 3 个分量的值改为 2，实现此操作的 MATLAB 语句如下：

```
≫b=ones(1,6);
≫b(1,3)=2;
≫b
```

敲回车，得到

b=1　1　2　1　1　1

当然，下列语句也是对的，而且更简洁

```
≫b=ones(1,6);
≫b(3)=2;
≫b
```

敲回车，得到

b=1　1　2　1　1　1

接着我们谈谈矩阵内部的矩阵块的赋值问题。仍然以矩阵 a 为例：

$$a=\begin{matrix}1 & 2 & 3 & 4 & 5\\ 6 & 7 & 8 & 9 & 10\\ 11 & 12 & 13 & 14 & 15\\ 16 & 17 & 18 & 19 & 20\\ 21 & 22 & 23 & 24 & 25\end{matrix}$$

假如想把矩阵 a 中用椭圆圈起来的矩阵块用一个 2 行 3 列零矩阵 b 去替换，那么 MATLAB 语句如下：

```
≫b=zeros(2,3);
≫a(3:4,2:4)=b;
≫a
```

敲回车，得到

$$a=\begin{matrix}1 & 2 & 3 & 4 & 5\\ 6 & 7 & 8 & 9 & 10\\ 11 & 0 & 0 & 0 & 15\\ 16 & 0 & 0 & 0 & 20\\ 21 & 22 & 23 & 24 & 25\end{matrix}$$

请注意，如下的语句：

```
≫b=zeros(2,3);
≫a([3 4],[2 3 4])=b;
≫a
```

敲回车，也可以得到相同结果。

要把矩阵 a 的第2行的所有元素用一个1行5列的全1向量去替换，MATLAB语句如下：

```
≫b=ones(1,5);
≫a(2,:)=b;
≫a
```

敲回车，得到结果：

```
     1    2    3    4    5
     1    1    1    1    1
a=  11   12   13   14   15
    16   17   18   19   20
    21   22   23   24   25
```

下述语句也可以得到与上面语句相同的结果，只是不如上面的语句简洁：

```
≫b=ones(1,5);
≫a(2,1:5)=b;
≫a
```

如果要把矩阵 a 的第3列的元素用一个5行1列的全1列向量去替换，MATLAB语句如下：

```
≫b=ones(5,1);
≫a(:,3)=b;
≫a
```

敲回车，得到

```
     1    2    1    4    5
     6    7    1    9   10
a=  11   12    1   14   15
    16   17    1   19   20
    21   22    1   24   25
```

同样，下述语句也可以得到与上面语句相同的结果：

```
≫b=ones(5,1);
≫a(1:5,3)=b;
≫a
```

1.7　矩阵运算与函数

对于任何一种软件或语言，熟练使用的前提是知道该软件或语言处理对象是什么，能够对该对象执行什么样的操作。如 Word 以文字为处理对象，能对文字进行设置字号、字体和颜色等操作。Photoshop 以各种图像文件为操作对象，可以对各种格式的图像进行修剪、缩放和旋转等操作。MATLAB 以矩阵作为基本操作对象（向量和数字是矩阵的退化情形），能够对矩阵进行加、减、乘、除、乘方、转置、求逆、取指数、对数、正弦等各种数学函数的操作。

其它高级计算机语言一般不能直接对矩阵进行各种运算，需要编制特定的子程序或函数来实现。而 MATLAB 可以直接对矩阵完成各种运算。

从线性代数的知识中可以知道，两个矩阵之间存在加减运算和乘法运算，且遵循一定的规则，有关规则读者可以参考各种线性代数教材。MATLAB 中矩阵的加减运算和乘法运算分别用加号（＋）、减号（－）、星号（＊）实现。

考虑矩阵 $a=\begin{matrix}1 & 2 & 3\\ 4 & 5 & 6\end{matrix}$ 和矩阵 $b=\begin{matrix}7 & 8 & 9\\ 6 & 4 & 5\end{matrix}$

矩阵 a 与矩阵 b 的加法和减法在 MATLAB 中的表达是：

```
≫a=[1 2 3;4 5 6];
≫b=[7 8 9;6 4 5];
≫c_plus=a+b;            %矩阵加法
≫c_minus=a-b;           %矩阵减法
≫c_plus
≫c_minus
```

敲回车，得到

$$c_plus=\begin{matrix}8 & 10 & 12\\ 10 & 9 & 11\end{matrix}$$

$$c_minus=\begin{matrix}-6 & -6 & -6\\ -2 & 1 & 1\end{matrix}$$

考虑矩阵 $a=\begin{matrix}1 & 2 & 3\\ 4 & 5 & 6\end{matrix}$ 和矩阵 $d=\begin{matrix}7 & 6\\ 8 & 4\\ 9 & 5\end{matrix}$

矩阵 a 与矩阵 b 的乘法在 MATLAB 中的表达是：

```
≫a=[1 2 3;4 5 6];
≫d=[7 6;8 4;9 5];
≫c=a *d
```

敲回车，得到

```
c=  50   29
   122   74
```

在线性代数中，还存在矩阵的数乘运算，即用一个数字去乘以矩阵，MATLAB 中也存在数乘运算，并且运算规则与线性代数的规定相同。

我们考虑矩阵 $a=\begin{matrix}1 & 2 & 3\\ 4 & 5 & 6\end{matrix}$ 与数字 3.5 的数乘运算，MATLAB 语句如下：

```
≫a=[1 2 3;4 5 6];
≫c=3.5 *a
```

敲回车，得到

```
c=  3.5000    7.0000   10.5000
   14.0000   17.5000   21.0000
```

语句：≫c=a *3.5 也能得到相同结果。

向量是矩阵的特殊情形或退化情形，有两种非常重要的向量乘法，一种是向量的点积运算，一种是向量的叉积运算，这两种运算均能在 MATLAB 中通过函数实现。

MATLAB 中函数的使用规则是：把多个自变量用逗号隔开，然后用小括号（）将自变量括起来，然后在小括号（）前面写下函数名。

MATLAB 中向量的点积运算使用 dot 函数实现。

考虑行向量 a=[1 2 3] 与 b=[3 5 8] 点积，MATLAB 语句如下：

```
≫a=[1 2 3];
≫b=[3 5 8];
≫c=dot(a,b)
```

敲回车，得到：

c=37

MATLAB 中向量的叉积用函数 cross 实现。

考虑行向量 a=[1 2 3] 与 b=[3 5 8] 叉积，MATLAB 语句如下

```
≫a=[1 2 3];
≫b=[3 5 8];
≫c=cross(a,b)
```

敲回车，得到

c=1 1 -1

根据线性代数知识，矩阵之间没有除法，但对于矩阵的退化情形——数字，则是存在除法的，数字之间的除法用斜杠（/）表示。

下面是 MATLAB 中数字除法的例子：

```
≫a=100;
≫b=12;
≫c=a/b
```

敲回车，得到

c=8.3333

MATLAB 中还有一种特殊除法，用反斜杠（\）表示，这种除法用来求解线性方程组，在后面的有关求解线性方程组的专题，我们将专门讲解，

MATLAB 的乘方运算使用算符（^），当进行矩阵乘方运算时，矩阵必须是方阵，即行数和列数相同的矩阵。

考虑矩阵 $a=\begin{matrix}1 & 2\\3 & 4\end{matrix}$，求矩阵 a 的平方的 MATLAB 语句如下：

```
≫a=[1 2;3 4];
≫a2=a^2;
```

运算结果如下：

≫a2

$a2=\begin{matrix}7 & 10\\15 & 22\end{matrix}$

MATLAB 中还存在一类特殊点运算，具体是点乘（.*），点除（./），点乘方（.^），这三种运算用于两个相同维度的矩阵之间，即进行点运算的两个矩阵应该具有相同的行数和列数。

两个相同大小矩阵进行点乘运算（.*）的规则如下：

考虑矩阵 $A=\begin{matrix}a_1 & b_1\\c_1 & d_1\end{matrix}$ 和矩阵 $B=\begin{matrix}a_2 & b_2\\c_2 & d_2\end{matrix}$ 的点乘（.*）

$$A.*B=\begin{matrix}a_1*a_2 & b_1*b_2\\c_1*c_2 & d_1*d_2\end{matrix}$$

两个相同大小矩阵进行点除运算（./）的规则如下：

考虑矩阵 $A=\begin{matrix}a_1 & b_1\\c_1 & d_1\end{matrix}$ 和矩阵 $B=\begin{matrix}a_2 & b_2\\c_2 & d_2\end{matrix}$ 的点除（./）

$$A./B=\begin{matrix}a_1/a_2 & b_1/b_2\\c_1/c_2 & d_1/d_2\end{matrix}$$

矩阵进行点乘方运算（.^）的规则如下：

考虑矩阵 $A=\begin{matrix} a_1 & b_1 \\ c_1 & d_1 \end{matrix}$ 的点乘方（.^）

$A.\hat{}\mathrm{n}=\begin{matrix} a_1\hat{}n & b_1\hat{}n \\ c_1\hat{}n & d_1\hat{}n \end{matrix}$

矩阵 A 还有另外一种有趣的点乘方运算，如下：

$\mathrm{n}.\hat{}A=\begin{matrix} n\hat{}a_1 & n\hat{}b_1 \\ n\hat{}c_1 & n\hat{}d_1 \end{matrix}$

上面我们讲解了用于矩阵之间的各种运算符，下面我们要谈一谈对矩阵取各种数学函数，如三角函数、对数函数等，还有一些其它的有用函数。

对一个矩阵取数学函数，比如取正弦函数，最后得到与原矩阵同样维度的，对这个矩阵的各个元素分别取正弦的矩阵。

比如对于矩阵 $A=\begin{matrix} a & b \\ c & d \end{matrix}$

假设要对矩阵 A 取正弦函数 sin，如下：

$\sin(A)=\begin{matrix} \sin(a) & \sin(b) \\ \sin(c) & \sin(d) \end{matrix}$

MATLAB 中的正弦函数是 sin，余弦函数是 cos，自然对数函数的 log，以 10 为底的对数函数是 log10，我们看一下对矩阵 a 取正弦函数的例子：

$\mathrm{a}=\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}$

```
≫a=[1 2;3 4];
≫a1=sin(a)
```

计算结果如下

$\mathrm{a1}=\begin{matrix} 0.8413 & 0.9093 \\ 0.1411 & -0.7568 \end{matrix}$

请注意，MATLAB 中三角函数接受的自变量输入值采用弧度而不是角度，反三角函数返回的数值是弧度而不是角度。

下表列出了 MATLAB 中的常用的数学函数，见表 1-1。

现在我们讨论一下矩阵的其它常用操作函数。

矩阵转置：采用算符单引号（'）对矩阵进行转置操作。若矩阵 A 的元素为实数，则与线性代数中矩阵的转置相同。若 A 为复数矩阵，则 A 转置后的元素由 A 对应元素的共轭复数构成。

对矩阵 $a=\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}$ 进行转置操作的 MATLAB 语句是：

表 **1-1**　**MATLAB** 中的常用的数学函数

函数类型	符　号	意　义
三角函数	sin	正弦
	asin	反正弦
	cos	余弦
	acos	反余弦
	tan	正切
	atan	反正切
	cot	余切
	acot	反余切
指数与对数函数	exp	以 e 为底的指数
	log	自然对数
	log10	常用对数
其它函数	abs	绝对值
	sqrt	平方根

```
a=[1 2; 3 4];
b=a'
```

结果是：

$$b=\begin{matrix}1 & 3\\ 2 & 4\end{matrix}$$

求和函数 sum：对矩阵按列求和，得到一行向量。如果被操作变量是一个向量，则将向量各个分量相加，获得向量各分量之和。对矩阵 $a=\begin{matrix}1 & 2 & 3\\ 4 & 5 & 6\end{matrix}$ 进行求和操作的 MATLAB 语句是：

```
≫a=[1 2 3;4 5 6];
≫b=sum(a)
```

结果是：

b=5　7　9

对于向量 a=[1 2 4 5] 进行求和操作的 MATLAB 语句是：

```
≫a=[1 2 4 5];
≫b=sum(a)
```

结果是：

b=12

矩阵求逆：使用 inv 函数求解。对矩阵 $a=\begin{matrix}1 & 2\\ 3 & 4\end{matrix}$ 求其逆矩阵的 MATLAB 语句是：

```
≫a=[1 2;3 4];
≫b=inv(a)
```

结果是：

$$b=\begin{matrix}-2.0000 & 1.0000\\ 1.5000 & -0.5000\end{matrix}$$

行列式的计算：使用det函数求解。对矩阵 $a=\begin{matrix}1 & 2\\ 3 & 4\end{matrix}$ 求其行列式值的MATLAB语句是：

```
≫a=[1 2;3 4];
≫b=det(a)
```

结果是：

b=-2

矩阵维度判断：size函数返回矩阵的行数和列数。对于矩阵 a，size(a)返回一二分量向量，这两个分量分别是矩阵 a 的行数和列数；size(a，1)返回矩阵 a 的行数，size(a，2)返回矩阵 a 的列数。对矩阵 $a=\begin{matrix}1 & 2\\ 3 & 4\end{matrix}$，其维度判断语句为：

```
≫a=[1 2;3 4];
≫n=size(a)                    %矩阵a的行数和列数
≫n1=size(a,1)                 %矩阵a的行数
≫n2=size(a,2)                 %矩阵a的列数
```

结果是：

n=2　2

n1=2

n2=2

向量长度函数：使用length函数返回向量的分量个数。对于向量 b=[1 3 6 7 9]，求向量 b 的分量个数的MATLAB语句如下：

```
≫b=[1 3 6 7 9];
≫L=length(b)
```

结果如下：

L=5

求矩阵的秩：使用rank函数求解。对矩阵 $a=\begin{matrix}1 & 2\\ 3 & 4\end{matrix}$ 求秩的MATLAB中的语句是：

```
≫a=[1 2;3 4];
≫r=rank(a)
```

结果是：

r=2

求向量的范数：一般向量的 2-范数比较常用，求向量 2-范数采用 norm(x) 格式的 MATLAB 函数，向量的 2-范数也是向量的长度。如求向量 a=[1 3 5 2] 的 2-范数的 MATLAB 语句是：

```
≫a=[1 3 5 2];
≫b=norm(a)
```

结果是：

b=6.2450

以上仅列出最常用的 MATLAB 中对矩阵和向量进行操作的函数，这和 MATLAB 中的全部函数比起来，实在是九牛一毛。读者在以后的科研与教学实践中，可能需要对数据执行某种操作，但并不知道 MATLAB 是否有这样的函数来完成上述操作，在这种情况下，读者可以利用网络上搜索引擎、MATLAB 官方网站的在线支持功能或 MATLAB 本身自带联机帮助文档的来查找是否有读者需要的函数。

如读者想求矩阵的特征值，想知道 MATLAB 是否提供这样的函数。方法之一是利用网络上的搜索引擎，打入搜索关键词“MATLAB”和“矩阵特征值”，搜索有关信息，能够发现利用 MATLAB 求特征值的函数是 eig。方法之二就是利用 MATLAB 官方网站的在线支持功能或 MATLAB 本身自带联机帮助文档，它们均提供搜索功能（Search），在搜索栏内输入特征值的英文“eigenvalue”，点搜索，从而得到 MATLAB 求特征值的函数是 eig，并能获得 eig 函数如何使用的详细信息。

此外，我们仅讲述了以上函数最主要的使用方式，但实际上 MATLAB 中的函数功能非常丰富。如何获得某函数的信息，从而挖掘某个函数的全部功能，使得该函数适应读者的需要，是一个很重要的问题。方法之一就是利用 MATLAB 的 help 命令，获得函数的比较详细的信息。如上面我们介绍了 norm 函数用于求向量的 2-范数，但我们还想知道有关 norm 函数的更多信息，这样，我们可以求助于 help 命令，格式如下：

≫help norm

敲回车，得到

```
NORM   Matrix or vector norm.
   For matrices...
      NORM(X) is the largest singular value of X,max(svd(X)).
      NORM(X,2) is the same as NORM(X).
```

```
    NORM(X,1) is the 1-norm of X,the largest column sum,
                    = max(sum(abs(X))).
    NORM(X,inf) is the infinity norm of X,the largest row sum,
                    = max(sum(abs(X'))).
    NORM(X,'fro') is the Frobenius norm,sqrt(sum(diag(X'*X))).
    NORM(X,P) is available for matrix X only if P is 1,2,inf or 'fro'.
  For vectors...
    NORM(V,P)=sum(abs(V).^P)^(1/P).
    NORM(V)=norm(V,2).
    NORM(V,inf)=max(abs(V)).
    NORM(V,-inf)=min(abs(V)).
  See also cond,rcond,condest,normest.
  Reference page in Help browser
     doc norm
```

如此，能够获得 norm 函数比较详细的信息，包括它的作用和使用方法。

方法之二是利用 MATLAB 官方网站的在线支持功能或 MATLAB 本身自带联机帮助文档，它们均提供搜索功能（Search），在搜索栏内输入"norm"，点搜索，可以得到关于 norm 函数更详细的信息。

1.8 MATLAB 的结构化程序设计方法与流程控制语句

在掌握了矩阵的定义、引用、赋值、运算符与各种矩阵操作函数后，我们已经能够利用 MATLAB 解决很多科学计算问题。但对于有些较为复杂的科学计算、数据分析与作图任务，我们必须把原始输入数据、各种赋值语句和各种操作函数有机地组织起来，使之成为一个系统的整体，顺利完成我们的各种任务。这样一种有机的组织体，称为程序。

1969 年荷兰科学家 E. W. 迪克斯特拉（E. W. Dijkstra）等人认真分析了人类面对复杂程序所表现出来的智力局限性后，认为现在的程序设计是一种控制巨大数据与指令并使其井然有序的复杂技术，是对人类智慧的一种严峻挑战，进而提出了一种新的程序设计方法——结构化程序设计。在解决一个复杂问题的时候，人的智力往往不可能一下触及问题的本质和细节，很难形成一个具体解决的方案。如果把给定的问题进行适当的分解，将问题的最终目标"要做什么"作为第一层或顶层，把完成这个问题的所有子问题作为第二层。然后再分析第二层，若能分析出更小的子问题，可作为第三层，如此进行下去，直到将问题分解为十分简单的、能由计算机通过很简单的操作指令就可实现如何"做"为止。经过这样处理后，一个极其复杂的问题，就转化为若干个十分简单的操作，就可以形成一个十分完美、具体的程序。

上述方法要求将程序分解成若干个相对独立、功能单一的子模块，并利用这些子模块组合成所需要的全局程序，这样一种方法称为模块化。模块化编写出来的程序就犹如一个由许多积木按照某种特定方式拼搭而成的金字塔，而那些积木就是一个个具有确定功能的子模块。如图 1-5 所示，矩形块表示各个模块，矩形块内的文字表示按照功能定义的模块名称，而矩形块之间的流向线则表示模块之间的调用关系。显然，整个程序按照调用关系分成若干层次，而每一层次由若干模块组成。这种方法的优点在于：①使得复杂的软件研制工作化整为零，便于群体分工与协作，从而极大地缩短了软件开发周期，节省了开发费用，提高了软件质量；②可以设计一块，调试一块，设计完成，调试也随之完成，模块的错误只存在于本模块内部，从而有效地杜绝了模块间错误的扩散；③整体程序结构灵活，层次分明，便于修改和维护，条理清晰，容易阅读和理解；④可以根据实际问题的需要对各模块进行适当组装，迅速产生新的软件系统。随着程序规模的不断扩大，其优越性愈显突出。

模块化方法的示意图见图 1-5。

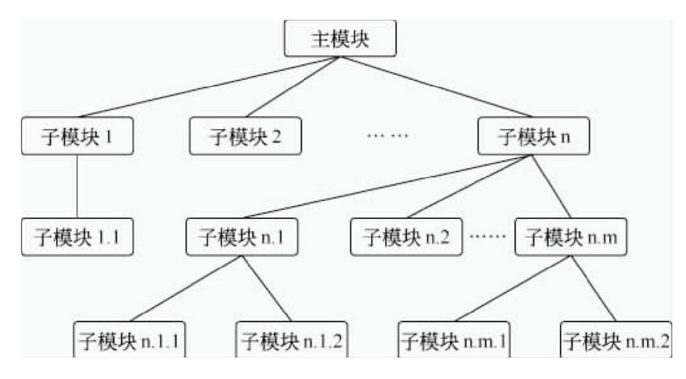


图 1-5　模块化方法

利用模块化方法将程序分解成若干个相对独立、功能单一的子模块后，需要将这些子模块组织成为所需要的全局程序。其次，在每个子模块内部，也需要将各种数据和指令有机的组织起来，建立子模块本身。这就提出了按照结构化方法在模块之间和模块内部进行组织，最终完成全局程序的编写。

1966 年 C. 玻姆（C. Bohm）指出，任何程序只要有一个出口和入口，且无死循环，均可通过顺序、选择、循环三种基本结构组合而成。顺序结构处理模式是按照各语句排列的先后顺序，依次逐语句执行，中间无分支或回转、跳转的情况。本书前面所举的用 MATLAB 计算需要取多少克 $K_2Cr_2O_7$ 的例子，就是一个顺序结构的例子；选择结构是存在多种分支，通过对实际问题的逻辑关系进行判断，执行某一特定分支。如日常生活中根据不同的天气状况而选择不同的交通出行工具，就是运用选择结构的一个例子；循环结构就是反复执行相同的操作，达到一定条件后终止。如砌墙就是反复的垒砖块，垒砖块这一动作被反复执行，但垒砖块到一定高度

后，则停止垒砖块的动作，执行其它操作，如将墙面刷上涂料。只有一个入口和出口，且无死循环，仅由顺序、选择、循环三种基本结构组合而成的程序称为结构化程序。完全按照结构化程序设计的三种基本结构，即顺序、选择和循环结构的流程图，见图 1-6。



图 1-6　结构化程序设计的三种基本结构的流程图

将一个复杂程序采用模块化方法分解成若干个相对独立、功能单一的子模块，然后利用三种基本结构将模块以及模块内部的数据和指令组织起来的编程方法，称为结构化的程序设计方法。

结构化程序设计方法的精髓在于模块化与结构化编码。

将一个复杂程序模块化的一般步骤是：①必须清楚地写出程序设计的总体构想和最终目标；②确定程序的总体结构，即将程序划分为若干子模块；③概括说明各子模块的功能，以及该模块与程序其它部分或其它模块之间的关系；④确定程序变量以及数据结构，并详细写出各个模块的具体操作步骤；⑤对每一模块进行预演，杜绝一切死循环和冗余，并最大限度地使用操作系统或编译系统提供的各种函数、控制符等，使程序结构和语句尽可能简单。

结构化的编码指利用顺序、选择和循环三种结构将各模块和模块内部的语句组织起来，我们现在的任务就是看一看在 MATLAB 中如何实现这三种结构。

并没有专门的 MATLAB 语句来实现顺序结构，只需要按照解决问题的逻辑顺序，依次书写 MATLAB 语句即可。

但存在实现选择结构和循环结构特定 MATLAB 语句。从图 1-6 可以直观地看出，无论是选择结构，还是循环结构，均需要根据条件 P 来执行。对于选择结构，当条件 P 为真（成立）时，执行 A 块，当条件 P 为假（不成立）时，执行 B 块。对于循环结构，当条件 P 为真（成立）时，执行循环体，当条件 P 为假（不成立）时，则跳过循环体，执行下面的语句。因此，我们需要知道 MATLAB 如何判断条件 P 的真假（成立或不成立）。

除了普通的数学四则运算、超越运算（指对数字取各种函数运算，如三角函数、指数或对数等）和分析运算（积分和微分）外，MATLAB 还支持关系运算和逻辑运算，关系运算比较两个数值的大小或两个数值是否相等，逻辑运算则可以将关系运算组合起来。如其它运算一样，关系运算和逻辑运算也会得到运算结果，关系运算和逻辑运算的运算结果是数值 1 或 0，1 代表运算结果为真，0 代表运算结果

为假。

关系运算是将数值通过关系运算符进行比较，得到运算结果 1 或 0（真或假，成立或不成立）。表 1-2 列出了 MATLAB 中的关系运算符。

表 **1-2　MATLAB** 中的关系运算符

关系运算符	说　明	关系运算符	说　明	关系运算符	说　明
＜	小于	＞	大于	＝＝	等于
＜＝	小于或等于	＞＝	大于或等于	～＝	不等于

下面我们看几个 MATLAB 关系运算的例子：

≫p＝1＞3

p＝0

≫p＝6.1＞＝3

p＝1

≫p＝1＜3

p＝1

≫p＝6＝＝5

p＝0

一般来说，选择或循环结构的很多条件 P 都表现为比较两个数值的大小或判定两个数值是否相等。一个单一的比较运算就能判断条件 P 的真假。

但有时条件 P 要由多个关系运算来确定，这时就需要用逻辑运算将多个关系运算组合起来，形成条件 P。我们在前面知道，关系运算的结果是 1 或 0，这些 1 或 0 参加逻辑运算后，其结果仍然是 1 或 0，可以判断条件 P 的真假。表 1-3 列出了 MATLAB 中的逻辑运算符。

表 **1-3　MATLAB** 中的逻辑运算符

逻辑运算符	运算规则说明	逻辑运算符	运算规则说明	逻辑运算符	运算规则说明
&	逻辑与，有 0 得 0，全 1 得 1	|	逻辑或，有 1 得 1，全 0 得 0	～	逻辑非，1 得 0，0 得 1

逻辑与和逻辑非运算符是对两个值进行逻辑运算，得到结果 1 或 0，而逻辑非仅仅对一个数值进行运算，得到结果 1 或 0。

我们下面看一下 MATLAB 逻辑运算的例子，体验上面讲述的运算规则：

逻辑与

≫p＝0 & 0

p＝0

≫p＝0 & 1

p＝0

≫p＝1 & 0

p＝0

≫p＝1 & 1

p＝1

逻辑或

≫p＝0|1

p＝1

≫p＝1|1

p＝1

≫p＝1|0

p＝1

≫p＝0|0

p＝0

逻辑非

```
≫p=~1                                  ≫p=~0
p=0                                    p=1
```

我们考察几个 MATLAB 中关系运算与逻辑运算组合，构成条件 P 的例子：

```
≫x=5;                                  ≫y=10;
≫p=(x>1)&(x<10)                        ≫p=(y>1)&(y<12)|(y==9)
p=1                                    p=1
≫z=6;                                  ≫w=11;
≫p=~(z>1)&(z<12)|(z~=9)                ≫p=~(w>3)&(w<15)|(w==9)
p=1                                    p=0
```

通过观察上面几个例子，我们可以知道当逻辑与&、逻辑或 | 和逻辑非~共同使用时，存在运算优先级问题，MATLAB 规定逻辑非~具有最优先运算级别，其次是逻辑与&，最后是逻辑或 | 。

MATLAB 中普通数学运算、关系运算与逻辑运算的优先顺序是：

普通数学运算>关系运算>逻辑运算

MATLAB 中的普通数学运算遵循通常的数学运算顺序，关系运算的各运算符具有相同的优先运算顺序。如果读者在编写 MATLAB 运算式子的过程中，忘记了各种运算的运算顺序，可以使用圆括号（ ）将希望优先运算的式子括起来，这样就能保证不混淆运算级别，防止错误发生。

在了解如何使用关系运算和逻辑运算构成条件 P 后，我们就可以学习 MATLAB 中特定的、能够构成选择结构和循环结构的语句。

MATLAB 中构成选择结构的最基本的语句是：

```
if  条件 P
    [条件 P 成立，执行本部分语句]
else
    [条件 P 不成立，执行本部分语句]
end
```

这是一个二分支选择结构。

我们举一个例子，来说明上面基本语句如何构成选择结构。假设如果购买苹果的数量在 6 斤以下，苹果的单价是 1.5 元/斤，如果购买苹果在 6 斤或以上，苹果的单价是 1.2 元/斤，现在想购买 3 斤苹果，需要多少钱？如果购买 10 斤苹果，需要多少钱？实现计算的 MATLAB 语句如下：

```
x=3;                          x=10;
if x<6                        if x<6
   money=x *1.5                  money=x *1.5
else                          else
```

```
    money=x *1.2              money=x *1.2
  end                       end
```

请注意，无须在MATLAB环境中逐个输入上面的语句，可以在记事本（Windows附件里自带的一个文本文件编辑软件）里书写上述语句，将这些语句复制，然后粘贴到MATLAB里，敲击回车，则MATLAB将执行这些语句，给出计算结果。计算结果分别是money=4.5000和money=12。

许多科学计算问题可能需要用到超过二分支选择结构的多分支选择结构，MATLAB同样提供了多分支选择结构的语句。

在多分支结构中，应该把出现几率较高的条件放在前面，当需要多次执行多分支结构时，这样安排，语句执行效率非常高。

```
if 条件 P1
   [如果条件 P1 成立，执行本部分语句]
elseif 条件 P2
   [如果条件 P2 成立，执行本部分语句]
elseif 条件 P3
   [如果条件 P3 成立，执行本部分语句]
   …
elseif 条件 Pn
   [如果条件 Pn 成立，执行本部分语句]
else
   [如果条件 P1，P2，P3，…Pn 都不成立，执行本部分语句]
end
```

如果在上面的多分支结构中，条件P_1和P_2都成立，是不是条件P_1部分的语句和条件P_2部分的语句都被执行呢？答案是否定的，MATLAB将优先执行条件P_1部分的语句而不执行P_2部分的语句，因为条件P_1部分在条件P_2部分之前，MATLAB仅仅执行多分支结构的一个分支。

使用多分支选择结构的一个典型例子就是计算分段函数的函数值。考虑如下分段函数：

$$Y=\begin{cases}\sin x & x\leqslant -10\\ \cos x & -10<x\leqslant 6\\ \sqrt{x-6} & x>6\end{cases}$$

计算该函数在任意一点函数值的问题，显然需要判断自变量x位于哪个区间，然后按照相应的自变量区间选择适当的函数解析式，然后进行函数值的计算。这个计算问题可以用一个多分支选择结构的语句来实现。比如想计算$x=-15$、$x=3$和$x=16$处的函数值，MATLAB语句以及计算结果如下：

```
x=-15;
if x<=-10
    y=sin(x)
elseif x>-10 & x<=6
    y=cos(x)
else
    y=sqrt(x)
end
y=-0.6503
```

```
x=3;
if x<=-10
    y=sin(x)
elseif x>-10 & x<=6
    y=cos(x)
else
    y=sqrt(x)
end
y=-0.9900
```

```
x=16;
if x<=-10
    y=sin(x)
elseif x>-10 & x<=6
    y=cos(x)
else
    y=sqrt(x)
end
y=4
```

上面的二分支选择结构或多分支选择结构存在退化情形，即选择结构的 else 部分可以省略。

对二分支结构，具体的退化情形是：

if　条件 P
　　[条件 P 成立，则执行本部分语句]
end

对于二分支选择结构而言。如果条件 P 为真，则执行选择结构内部的语句，如果条件 P 不成立，则忽略该选择结构，执行该选择结构后面的语句。

对多分支结构，退化情形是：

if　条件 P_1
　　[如果条件 P_1 成立，执行本部分语句]
elseif 条件 P_2
　　[如果条件 P_2 成立，执行本部分语句]
elseif 条件 P_3
　　[如果条件 P_3 成立，执行本部分语句]
　　…
elseif 条件 P_n
　　[如果条件 P_n 成立，执行本部分语句]
end

对于多分支选择而言，如果条件 P_1，P_2，P_3，…，P_n 有一个或多个为真，则仅执行条件为真且位于最前面的那个条件所对应的语句，如果所有条件都不成立，则忽略该选择结构，执行该选择结构后面的语句。

在学习了 MATLAB 的选择结构后，我们接着讲述 MATLAB 中的循环结构。如前面所讲，所谓循环，就是反复做同一件事情，同时还需要设置一个条件 P，使得条件 P 得不到满足时停止循环，从而继续执行循环体后面的语句，避免无休止的循环或死循环。表面看来，反复做同一件事情好像毫无意义，对事情的进展并没有帮助。但实际上，我们的日常生活中做的大量事情都是循环，比如盖房子，需要反

复把砖块垒上去，走路需要不断地、反复地迈动双腿。盖房也好，走路也好，虽然都是反复在做同样的事情，但每做一次，都是对要达到的目标的一个促进或积累，直到完成任务为止。

科学计算问题存在大量需要使用循环结构的问题，MATLAB 存在两种语句来实现循环结构，一种语句用于已经知道循环的次数，另外一种语句用于循环次数事先未知，需要用条件 P 来确定的情况。

考虑一个著名的加法问题，求和 $S=1+2+3+\cdots+99+100$，天才的数学家高斯在小时候通过敏锐的观察，变加法为乘法，迅速而正确的得到答案：5050。这实际是一个等差数列的求和问题，可以通过等差数列的求和公式计算，但假设我们没有等差数列的概念，希望通过计算机程序解决这个问题，该如何去做呢？从最直观的角度看，我们要求这个和，采用的加法格式是：

$$S=S+k$$

开始时 $S=0$，然后 k 分别取 1、2、3、…、99、100 加到 S 上面，S 的大小在加法过程中不断增加，直到 k 最后取 100 加到 S 上面，完成求和。实际上做了 100 次加法。100 次加法？可以使用循环结构来实现！因为循环结构就是反复做同一件事情，我们这里是反复做加法，做 100 次。

首先的问题是，如何让 k 依次取 1、2、3、…、99、100，其次的问题是如何让不同的 k 值依次加到 S 上面去，即做 100 次加法。

上面两个问题 MATLAB 中的实现方法是采用 for…end 循环语句：

for…end 循环的一般形式是：

```
for  k=向量
    [for 循环体内语句]
end
```

向量包含有多少个元素，[for 循环体内语句] 就被反复执行多少次，在第 m 次循环中，变量 k 被赋值为向量的第 m 个分量。

根据以上分析，可以写出如下的 MATLAB 语句：

```
S=0;
for k=1:100
    S=S+k;
end
S
```

运行，得到 S=5050

1:100 是以 1、2、3、…、99、100 为分量，长度为 100 的向量。通过 for…end 循环语句，k 依次取 1、2、3、…、99、100，依次加到 S 上，最后得到和数 5050。

有不少科学计算需要使用事先不知道循环次数的循环结构来解决，对于这种结

构，MATLAB使用while…end语句来实现。

```
while 条件P
    [当条件P为真时，执行循环体内语句]
end
```

我们举一个使用while…end语句的例子。

验证角谷猜想：对于任意一个自然数，若为偶数，把它除以2；若为奇数，把它乘以3后再加1。如此经过有限次运算后，最终能得到自然数1。（角谷猜想是一个世界性难题，由日本数学家角谷静夫首先提出。）

很明显，用计算机验证角谷猜想需要用到循环结构，当自然数不等于1时，对自然数反复进行除以2或乘以3后再加1的操作。在写出验证角谷猜想的MATLAB语句之前，我们需要知道，MATLAB中是否存在函数判断一个数是否是偶数或奇数，这可以通过MATLAB的rem函数来实现，rem函数的参数是两个整数。rem（M,N）的作用是求整数M除以整数N得到的余数，这样我们就可以通过rem（M,2）是否等于0来判断M是偶数还是奇数。

以自然数12为例子，验证角谷猜想的MATLAB语句如下：

```
M=12        %以自然数12为例子
N=0;        %N是计数变量,统计最终得到自然数1后,一共循环了多少次
while M~=1       % 执行的循环条件
    if rem(M,2)==0
        M=M/2       %为偶数,把它除以2
    else
        M=3*M+1       %为奇数,把它乘以3后再加1
    end
    N=N+1;   %每执行循环1次,N就增加1,以统计共执行多少次循环操作
end
N       %输出N
```

运行，结果如下：

```
M=12                M=16
M=6                 M=8
M=3                 M=4
M=10                M=2
M=5                 M=1
                    N=9
```

上述MATLAB语句给出每次循环运算得到的M值，经过9次循环后，M变为1。

选择结构和循环结构语句被称为流程控制语句，因为它们均根据一定条件来控制

程序语句的执行方向或执行状况。流程控制语句在计算机语言中具有极其重要的地位，因为它们是组织指令与数据、构成程序的基本单元。能不能将孤立的语句组织起来，写出有机的、能够有效解决问题的程序，关键取决于能否灵活运用流程控制语句。

1.9　MATLAB 的函数文件与脚本文件

结构化程序设计方法要求把一个大型程序模块化，然后对各层次子模块加以组织，整合成完整的程序。在 MATLAB 中，子模块在形式上表现为 MATLAB 的函数，各子模块经过组织后得到的完整程序表现为 MATLAB 的脚本文件。

我们先来了解 MATLAB 的函数。MATLAB 的函数分为内部函数和外部函数。内部函数是 MATLAB 本身自带的函数，如前面提及的矩阵求逆函数 inv，求和函数 sum，正弦函数 sin 等。外部函数是人们为了计算工作的方便，自己编写的函数。MATLAB 附带的各种工具箱内的函数，还有其它科研工作者为本科研领域工作方便自己编写工具箱内的函数，都是实现特定功能的 MATLAB 外部函数集合。

外部函数和内部函数一样，都具有函数名和自变量，外部函数和内部函数在 MATLAB 环境中使用的方法也是一样的，都是把多个自变量用逗号隔开，然后用小括号（ ）将自变量括起来，然后在小括号（ ）前面写下函数名。

MATLAB 外部函数一般可以通过函数文件进行定义，定义格式如下：

function$[y_1, y_2, \cdots, y_{n-1}, y_n] = f(x_1, x_2, \cdots, x_{n-1}, x_n)$

　　[函数本体包含的各种语句]

定义外部函数，关键字 function 是必须有的，且要写在外部函数第一行的首位。字符 f 是函数名，它代表定义的外部函数的名字，用小括号（ ）括起来的 x_1，x_2，…，x_{n-1}，x_n是外部函数的自变量，即函数的输入值，用中括号 [ ] 括起来的 y_1，y_2，…，y_{n-1}，y_n是函数的因变量，也就是函数的输出值，MATLAB 的函数允许函数有多个输出值。

函数比通常的一段程序语句具有大得多的灵活性，比如我们前面编写过求 $S=1+2+3+\cdots+99+100$ 之和的程序以及对自然数 12 验证角谷猜想的程序，这两个程序都是对的，但不具有普遍性，也许我们想求 $S=1+2+3+\cdots+999+1000$ 的和，我们可能想对 100、999、1000 来验证角谷猜想，这样我们就需要对这两个程序做出某些改动（有时这种改动可能很烦琐），才能完成新的任务，这是一段程序语句的局限性所在，此外一段程序语句无法被其它函数或程序调用。但函数的特性在于不用改动函数内部的语句，只需要改变自变量的值，函数值就会相应改变，而且函数还能被其它程序或函数调用，这使得函数在编写程序中具有巨大的优越性，完全符合模块化、结构化的思想。

按照上面外部函数定义的规则，我们把求 $S=1+2+3+\cdots+99+100$ 和的程序以及对自然数验证角谷猜想的程序写成更通用的函数形式，如下：

求 S=1+2+3+…+(M−1)+M 的和

```
function S=qiuhe(M)
S=0;
for k=1:M
    S=S+k;
end
```

对任意自然数 M 验证角谷猜想

```
function N=jiaogu(M)
N=0;
while M~=1
if   rem(M,2)==0
          M=M/2
    else
          M=3 *M+1
    end
    N=N+1;
end
```

上述两个函数均只有一个输入值和一个输出值，但 MATLAB 允许函数有多个输入值和输出值，这完全视具体的情况而定。上述两个函数的函数名分别是 qiuhe 和 jiaogu，它们的地位和作用相当于 MATLAB 的三角函数的函数名，如 sin、cos 等。

在定义了外部函数后，该如何使用呢。以求 S=1+2+3+…+(M−1)+M 的和函数 qiuhe 为例：首先需要点击 MATLAB 的 File 菜单，然后点击 New 子菜单，然后再点击 M-file 项目，打开 MATLAB 的 M 文件编辑器 Editor，见图 1-7。

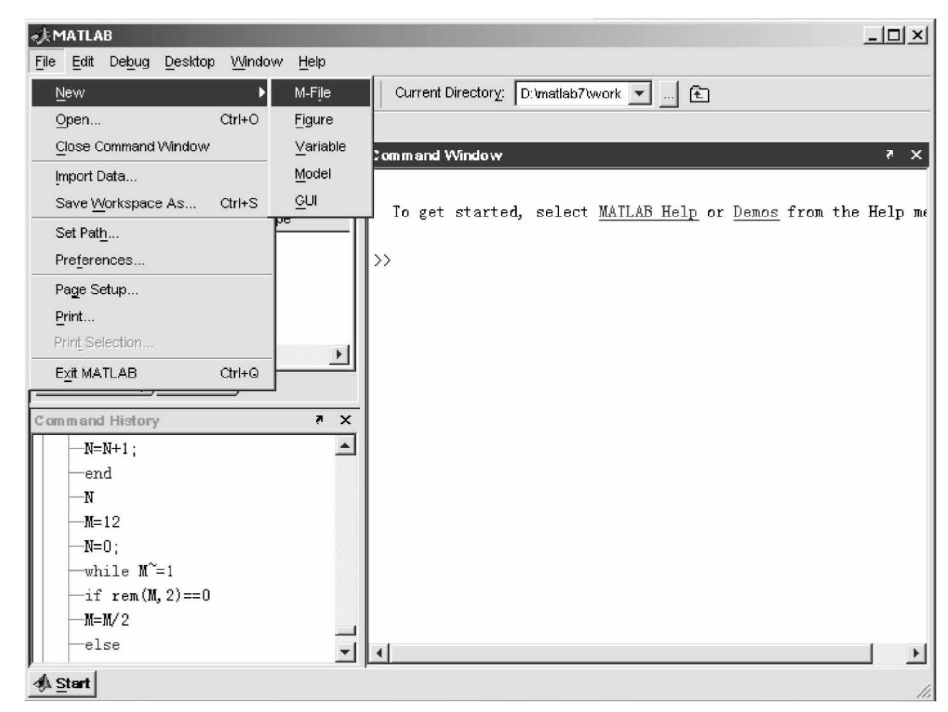


图 1-7　打开 MATLAB 的 M 文件编辑器 Editor

其次在打开的 M 文件编辑器输入 qiuhe 函数的代码，也可以在 windows 自带的记事本里输入 qiuhe 函数的代码，然后粘贴到 M 文件编辑器里。

最后点 M 文件编辑器 Editor 的 File 菜单，点其中的 save 项目，将 qiuhe 函数保存在 MATLAB 安装目录下的 work 子目录里，文件名取函数名 qiuhe，文件扩展名是 .m，见图 1-8。

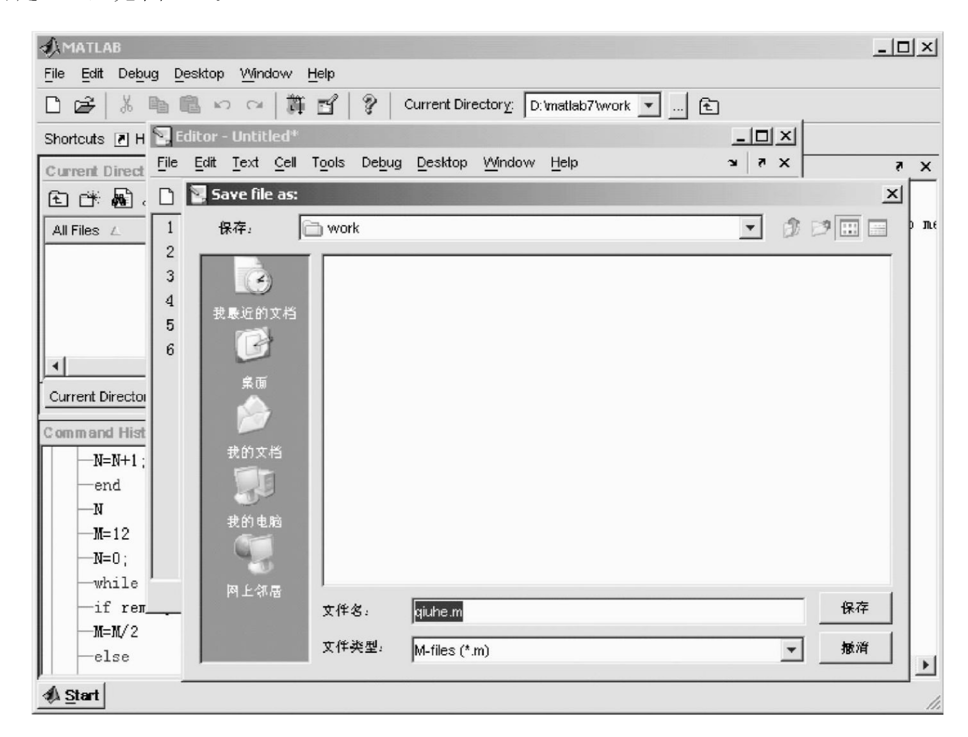


图 1-8　将 qiuhe 函数保存在 MATLAB 安装目录下的 work 子目录里，文件名 qiuhe. m

将外部函数 qiuhe 保存在 MATLAB 安装目录下的 work 子目录里，文件名 qiuhe. m 后，就可以像使用 MATLAB 内部函数一样来使用外部函数 qiuhe 了。

如想计算 $S=1+2+3+\cdots+200$ 和 $S=1+2+3+\cdots+300$ 的和，MATLAB 语句如下：

```
≫S=qiuhe(200)
S=20100
≫S=qiuhe(300)
S=45150
```

由此可以看出，使用 MATLAB 的函数具有多么巨大的优越性。

现在我们学习如何定义和使用 MATLAB 的脚本文件。实际上，MATLAB 的脚本文件就是一段按照结构化编码原则进行编制，包含数据，流程控制语句、函数和其它语句的代码集合。这段代码可以用 MATLAB 的 M 文件编辑器 Editor 进行

编辑书写（也可以用 windows 自带的记事本编辑书写，然后复制粘贴到 M 文件编辑器 Editor 中），然后将这段代码在 M 文件编辑器 Editor 的环境下存盘，保存为一个扩展名为 .m 的文件。就得到了一个 MATLAB 的脚本文件。

我们通过以位于本书第 2 页的计算需要应称取 $K_2Cr_2O_7$ 多少克为例，具体说明如何定义和使用脚本文件。

首先在 MATLAB 的 M 文件编辑器 Editor 编辑书写如下代码：

```
M=294.2;                    %输入K2Cr2O7的摩尔质量
C=0.02000;                  %输入K2Cr2O7标准溶液的浓度
V=250;                      %输入K2Cr2O7标准溶液的体积
m=C*V/1000*M                %计算需要的K2Cr2O7的质量,单位g
```

其次点 M 文件编辑器 Editor 的 File 菜单，点其中的 save 项目，将本段代码保存在 MATLAB 安装目录下的 work 子目录里，文件名取函数名 zhiliang（请注意脚本文件的取名是任意的，只要符合 MATLAB 系统的文件命名规则即可，但外部函数在存盘时，文件名必须和要定义的外部函数名一致），文件扩展名是 .m。

如此，就完成了 MATLAB 脚本文件的定义。

使用时，只需要在 MATLAB 环境中键入该脚本文件名：

```
>>zhiliang
```

点回车，得到：

```
m=1.4710
```

上述程序代码也可以用 windows 自带的记事本编辑书写，然后复制、粘贴到 MATLAB 的运行环境，直接敲回车运行，不必保存为脚本文件。

但对于大型程序来说，使用脚本文件具有很大的优越性，首先可以将程序代码永久保存，更重要的是，在脚本文件中，MATLAB 的各种函数、语句和数据被组织起来，形成一个有机的、好的、完整的程序，以解决大型的科学计算或数据处理问题。

1.10 MATLAB 的函数句柄

MATLAB 的函数句柄常用来快速定义通常意义上的数学函数。比如对数学函数：

$$f(x)=\sin(x)+\cos(x)$$

自然可以用前面讲述 MATLAB 函数文件进行定义，如下：

```
function y=f(x)
         y=sin(x)+sin(x);
```

将上面的函数文件起名为 f.m 然后存盘在 MATLAB 的安装目录下的 work 子目录中，就可以在 MATLAB 中使用函数 $f(x)=\sin(x)+\cos(x)$。

然而更方便的定义函数 $f(x)=\sin(x)+\cos(x)$ 的方法是使用函数句柄，在 MATLAB 环境下输入如下语句：

f=@(x)sin(x)+cos(x)；

这样就获得一个称为函数句柄的变量 f，通过上述定义，f 就代表了以 x 为自变量的函数 $\sin(x)+\cos(x)$，比如我们想求 sin(2)+cos(2)的值，可以写下如下代码：

```
f=@(x)sin(x)+cos(x);
b=f(2)
```

结果是：

b=0.4932

利用函数句柄，可以更方便更快速地定义普通的数学函数，是一种值得推荐的好方法，但对于分段函数或需要书写若干代码才能求得函数值的函数［如 S=1+2+3…+(M-1)+M 的和函数 qiuhe］还需要利用函数文件进行定义。

1.11　MATLAB 的复数、数据精度与常数

MATLAB 区别于其它高级计算机语言的一个特征之一就是 MATLAB 中可以直接使用复数。在其它高级计算机语言中，要么不能使用复数，需要科研工作者另外自己编写函数进行定义，要么采用特殊形式来处理复数，如专门为科学计算诞生的 Fortran 语言用一个二元数对（a，b）来定义复数，其中 a 代表复数的实部，b 代表复数的虚部，显然，这样处理复数，会为科学计算带来很大的不便，致使实数和复数无法共同参与运算。

我们从数学知识中知道，复数表示为 $a+b$i，其中 a 代表复数的实部，b 代表复数的虚部，i 是虚数单位。MATLAB 也遵循相同的定义复数的规则。i 在 MATLAB 中定义为虚数单位，复数表示为 $a+b$i。在 MATLAB 中，j 也被定义为复数单位，因此在 MATLAB 中，复数也可表示为 $a+b$j，但从习惯上考虑，一般复数写成 $a+b$i 形式比较符合大家的习惯。

我们看几个 MATLAB 复数运算的例子，来体验一下 MATLAB 处理复数是多么流畅自然，体现了数学的完美与统一，通过这些例子，大家能够真正体会到 MATLAB 确实是为科学计算诞生的软件和语言。

```
>>a=i^2
a=-1
>>a=sqrt(-4)
a=0+2.0000i
>>a=log(-18)
a=2.8904 + 3.1416i
>>a=(1+2i)*(3-5i)-2i
a=13.0000-1.0000i
>>a=(2+i)/(5+2i)+6
a=6.4138 + 0.0345i
>>a=sin(1+2i)
a=3.1658+1.9596i
```

MATLAB在进行科学计算时，数据运算精度很高，MATLAB默认的数据精度是所谓的双精度数据，MATLAB的双精度数具有15位有效数字，但我们从前面的运算结果看到，MATLAB运算结果通常只有5位有效数字。这是因为MATLAB默认将运算结果以所谓的short格式在屏幕上显示，为5位有效数字，但实际上，运算结果是15位有效数字，只是仅显示5位而已。要想让MATLAB显示15位有效数字，可以用format long命令将MATLAB的数据显示格式改为long格式，即显示15位有效数字。

举例如下：

```
≫a=log(5)
a=1.6094
≫format long
≫a
a=1.60943791243410
```

通过例子，我们可以看到，使用format long命令后，log(5)的值显示了15位有效数字。

如果想将数据的long显示格式改为short显示格式，可以用format short命令，举例如下：

```
≫b=sqrt(2)
b=1.41421356237310
≫format short
≫b
b=1.4142
```

MATLAB里还有一些数学常用的常数，这些常数是MATLAB内部定义好的，以方便科研人员进行科学计算。

最著名的数学常数莫过于圆周率P，MATLAB里圆周率P用pi表示：

```
≫a=pi
a=3.14159265358979
```

前面我们提到，虚数单位用i或j来表示：

```
≫a=i^2                    ≫a= j^2
a=-1                      a=-1
```

在分析运算中，有个著名常数无穷大∞，在MATLAB中用inf表示，在一般的高级计算机语言中，一个非零的常数去除以零是错误的、没有意义的，但MATLAB把这种情况处理为得到无穷大：

```
≫a=1/0
Warning:Divide by zero.
a=Inf
```

≫a＝－2/0

Warning: Divide by zero.

a＝－Inf

pi，i，j，inf 在 MATLAB 中分别代表不同的数学常数，因此希望读者在编写 MATLAB 程序时，不要使用名称为 pi，i，j，inf 的变量，不要对 pi，i，j，inf 进行赋值，如不要出现 pi＝12 这样的语句，以免引起混乱，因为 pi，i，j，inf 在 MATLAB 里具有特定的含义和数值。

另外，著名的数学常数 e 在 MATLAB 里如何表示呢？是不是 MATLAB 也采用了一个符号来定义了常数 e 呢？答案是没有！如果我们用到 e，那么该如何表示呢？我们前面学习过以常数 e 为底的指数函数是 exp(x)，因此，常数 e 通过 exp(1) 运算得到：

≫a＝exp(1)

a＝2.71828182845905

MATLAB 还有一个特殊常量，称为非数，它用 NaN 表示，在极限运算中，无穷大减无穷大、零除以零常常没有确定的极限，它们的值是不确定的，对于这些情况，MATLAB 用 NaN 来表示，如下：

≫a＝inf－inf

a＝NaN

≫b＝0/0

Warning: Divide by zero.

b＝NaN

1.12　使用 MATLAB 进行计算的一些注意事项

我们在前述的需要称取 $K_2Cr_2O_7$ 多少克的例子里，曾经提到赋值号（＝），赋值号（＝）的作用是将计算式的值赋给某个变量，这里说的计算式，指的是由括号、各种数学运算符（如加减乘除）和函数将各种数据（如常数和变量）组合起来的有意义的式子。赋值号的作用是将赋值号（＝）左边计算式的值赋给赋值号（＝）右边的变量，请大家注意，它绝对不同于数学上的等于号，我们前面也提到过，在 MATLAB 中，等于号是（＝＝），专门用来判断两个数值是否相等。

假设有两个变量 $a=3$，$b=4$；

现在有如下两个赋值语句：

a＝b；

b＝a；

这两个赋值语句的作用是截然不同的，第一个语句的作用是将 b 的值赋给 a，赋值后 a 和 b 的值都是 4。第二个赋值语句的作用是将 a 的值赋给 b，赋值后 a 和 b

的值都是 3，MATLAB 语句与运行结果如下：

```
a=3;                                  a=3;
b=4;                                  b=4;
a=b;                                  b=a;
a                                     a
b                                     b
a=4                                   a=3
b=4                                   b=3
```

赋值号左边只能是变量，不能是计算式子，例如：a+b=3 *5+1 是错误的。

考察下面的语句：

N=N+1

它的作用是将 N 的原值加 1 再送回 N 中去，如果 N 的原值为 5，那么通过上述语句，N 的值变成了 6。这个语句当然是正确的，原因就在于在 MATLAB 中，(=) 是赋值号，而不是数学里的等号。

MATLAB 还提供在运行环境中直接输入计算式子进行直接计算，而不需要用赋值号把该式子的值赋给其它变量。当 MATLAB 直接计算式子的值时，将式子的值默认赋值给变量 *ans*，变量 *ans* 是 MATLAB 的保留变量名，专门用于储存没有进行赋值的计算式子的值。

例如：计算$\sqrt[3]{5}\left(\cos\dfrac{4\pi}{3}\right)$的值。

MATLAB 语句如下：

```
>>5^(1/3)*cos(4*pi/3)
```

敲回车，得到

```
ans=-0.8550
```

我们看到，上面的计算式子没有赋值给任何变量，但 MATLAB 直接计算了该式子的值，并将该值赋给 MATLAB 的保留变量 *ans*。

我们前面对 $s=1+2+3+\cdots+99+100$ 的值，采用循环结构语句求解，在这里，我们采用另外一种方法求解，由此，读者也可体验到 MATLAB 的优越性所在。

求解语句代码如下：

```
s=sum(1:100)
```

敲回车，得到

```
s=5050
```

由于 MATLAB 能够直接表示和操作矩阵（含向量），因此，我们首先生成一个含 100 个元素、分量为 1、2、3、…、99、100 的行向量，然后用 sum 函数对该向量的各分量相加求和，只用一条语句就得到结果。

我们再看一个例子：

计算 $s=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots+\frac{1}{197}-\frac{1}{199}$的值

这个问题比求 $s=1+2+3+\cdots+99+100$ 的值要稍微复杂一些，但本质上仍然可以使用循环结构语句来解决，自然，也能利用 MATLAB 处理矩阵和向量的强大功能解决之。我们下面写出解决这个问题的两种方法的语句代码，读者可以比较一下。

循环语句方法

```
s=0;
p=-1;
for k=1:2:199
    p=-p;
    s=s+p/k;
end
s
```

向量方法

```
k=zeros(1,100);
k(1:2:99)=1:4:197;
k(2:2:100)=-1 *[3:4:199];
s=sum(1./k)
```

运行上述两种方法的语句代码，结果都是 s=0.7829。

对这两种方法的说明如下。

该求和序列每项的通项是 $(-1)^{n+1}\frac{1}{2n-1}$，只要能够把求和序列的通项表达出来，则该序列的求和问题就解决了。

循环方法首先定义了一个向量［1 3 5…197 199］，每循环一次，k 取该向量的一个分量，当循环次数是奇数时，p 取 1，当循环次数为偶数时，p 取－1，每次循环，就可以用 p/k 表示该求和序列的通项，然后利用循环体内的语句 s=s+p/k，逐步把该求和序列的每一项加到 s 上，当循环完成后，就求得该序列的和。

向量方法首先定义了一个含有 100 个零的行向量，后面将用这个向量存放求和序列的 100 个通项。这个求和序列的奇数项的分母是向量［1:4:197］，其偶数项的分母是－1 *［3:4:199］，然后将向量［1:4:197］赋给向量 k 的奇数分量，将向量－1 *［3:4:199］赋给向量 k 的偶数分量，这样，向量 k 的奇数项就是求和序列的奇数项的分母，其偶数项就是求和序列的偶数项的分母，这样，1./k 的各分量就是求和序列的各项（请注意这里使用了点除算符），然后用 sum 函数对 1./k 求和，得到序列和。

我们最后需要提及的是，MATLAB 是区分大小写的，也就是说，a 和 A 被 MATLAB 认为是两个不同的变量。对于 MATLAB 本体所带的各种语句、函数和命令，应该使用小写字母，否则 MATLAB 可能报错。如计算 3 的正弦值，使用 Sin(3)这个语句，MATLAB 将报错，应当使用小写的 sin(3)。

1.13 利用MATLAB进行简单计算

我们说的简单计算是指仅仅涉及四则运算（加减乘除）和超越运算（取幂、对数、指数、三角函数）的计算，简单计算既不涉及分析运算（微分和积分），也不涉及各种方程和方程组的求解。下面我们举两个例子，说明如何利用MATLAB进行简单计算。

1.13.1 反应的标准摩尔焓变

作为灵活使用向量点积函数的一个有趣例子，我们考察如何使用dot函数求解化学反应过程中热力学函数的变化。

对于化学反应：

$$v_1 A + v_2 B \longrightarrow v_3 C + v_4 D$$

反应物A、B反应生成产物C、D，现在要求该反应的某个标准热力学函数的变化，如该反应的标准摩尔焓变。

根据热力学有关知识，求该反应的标准摩尔焓变的式子可以写作：

$$\Delta_r H_m^\ominus = \sum_i v_i \Delta_f H_m^\ominus$$

对于上述反应，可以把反应物和生成物的计量系数组成为一个计量系数向量

$$[v_1\ v_2\ v_3\ v_4]$$

反应物的计量系数规定为负值，生成物的计量系数规定为正值。

反应物和生成物的标准摩尔生成焓相应的也可以组成一个标准摩尔生成焓向量：

$$[\Delta_f H_m^\ominus(A)\,\Delta_f H_m^\ominus(B)\,\Delta_f H_m^\ominus(C)\,\Delta_f H_m^\ominus(D)]$$

很明显，根据向量点乘的定义和上述求该反应的标准摩尔焓变的式子，计算反应的标准摩尔焓变实际上就是求反应的计量系数向量与标准生成焓向量的点积：

$$\Delta_r H_m^\ominus = \text{dot(计量系数向量,标准摩尔生成焓向量)}$$

例如求下列反应的标准摩尔生成焓 $\Delta_r H_m^\ominus$：

$$2Na_2O_2(s) + 2H_2O(l) \longrightarrow 4NaOH(s) + O_2(g)$$

反应的计量系数向量为［－2 －2 4 1］，反应的标准摩尔生成焓向量为［－513.2 －285.82 －426.73 0］，此处标准摩尔生成焓的单位是 $kJ \cdot mol^{-1}$。

计算该反应标准摩尔焓变的MATLAB语句如下：

```
>>a=[-2 -2 4 1];
>>b=[-513.2 -285.82 -426.73 0];
>>delta_H=dot(a,b)      %标准摩尔生成焓的单位是 kJ·mol^-1
```

敲回车，得到

```
delta_H=-108.8800
```

1.13.2　求解反应平衡常数

我们下面来看一个物理化学方面的问题。本问题的计算步骤较多，虽然本问题用计算器就可以完成，但利用计算器不容易检查计算步骤，输入数据和算式时容易出错。用 MATLAB 程序一段程序代码进行计算，并在每条语句后面加上注释性文字，则可最大限度地防止错误的发生，保证计算正确。

已知下列标准电极电势：

$Cu^{2+} + 2e^- \Longleftrightarrow Cu \qquad \varphi^{\ominus} = 0.337V$　①

$Cu^{2+} + e^- \Longleftrightarrow Cu^+ \qquad \varphi^{\ominus} = 0.159V$　②

（1）计算反应 $Cu(s) + Cu^{2+}(aq) \Longleftrightarrow 2Cu^+(aq)$ 的平衡常数（25℃）；

（2）已知 $K_{sp}^{\ominus}(CuCl) = 1.2 \times 10^{-6}$，试计算反应 $Cu(s) + Cu^{2+}(aq) + 2Cl^-(aq) \Longleftrightarrow CuCl(s)$ 的平衡常数（25℃）。

对于问题（1），如果能知道反应 $Cu(s) + Cu^{2+}(aq) \Longleftrightarrow Cu^+(aq)$ 的标准吉布斯自由能变，则可根据标准吉布斯自由能变与平衡常数的关系，得到反应的平衡常数。通过观察可知，反应②乘以 2 再减去反应①，就可得到反应 $Cu(s) + Cu^{2+}(aq) \Longleftrightarrow Cu^+(aq)$，根据热力学函数的变化与途径无关的原理，如果知道反应①和②的标准吉布斯自由能变，则反应②的标准吉布斯自由能变乘以 2 再减去反应①的标准吉布斯自由能变，就能知道反应 $Cu(s) + Cu^{2+}(aq) \Longleftrightarrow Cu^+(aq)$ 的标准吉布斯自由能变。根据标准电极电势与标准吉布斯自由能变的关系，反应①和②的标准吉布斯自由能变容易得到。因此，问题（1）解决。

对于问题（2），沉淀反应 $Cu^+ + Cl^- \Longleftrightarrow CuCl(s)$ 乘以 2 再加上反应 $Cu(s) + Cu^{2+}(aq) \Longleftrightarrow 2Cu^+(aq)$，就可得到反应 $Cu(s) + Cu^{2+}(aq) + 2Cl^-(aq) \Longleftrightarrow CuCl(s)$，因此，沉淀反应的标准吉布斯自由能变乘以 2 减去反应 $Cu(s) + Cu^{2+}(aq) \Longleftrightarrow 2Cu^+(aq)$ 的标准自由能变，就可求得反应 $Cu(s) + Cu^{2+}(aq) + 2Cl^-(aq) \Longleftrightarrow CuCl(s)$ 的标准吉布斯自由能变，根据标准吉布斯自由能变与平衡常数的关系，就可求出反应 $Cu(s) + Cu^{2+}(aq) + 2Cl^-(aq) \Longleftrightarrow CuCl(s)$ 的平衡常数。反应 $Cu(s) + Cu^{2+}(aq) \Longleftrightarrow 2Cu^+(aq)$ 的标准自由能变已经由问题（1）求得，现在又知道沉淀反应 $Cu^+ + Cl^- \Longleftrightarrow CuCl(s)$ 溶度积，根据溶度积，很容易求得沉淀反应的标准吉布斯自由能变。则问题（2）解决。

还有一个小问题就是溶度积常数是用科学计数法表示的，在 MATLAB 中如何表示？用科学计数法表示的数值，如 1.2×10^{-6}，在 MATLAB 中的表示方法是 1.2e-6。

解决上述问题的 MATLAB 语句如下：

```
%求解问题(1)
F=96485.34;                    %法拉第常数
T=25+273.15;                   %反应的热力学温度
R=8.314;                       %理想气体常数
delta_G1=-2*F*0.337;           %计算反应①的标准吉布斯自由能变
```

```
delta_G2=-1 *F *0.159;                  %计算反应②的标准吉布斯自由能变
delta_G_1=2 *delta_G2- delta_G1;        %计算反应(1)的标准吉布斯自由能变
K1=exp(delta_G_1/(-R *T))               %计算反应(1)的平衡常数
%求解问题(2)
Ksp=1.2e-6;                             %沉淀反应的溶度积常数
delta_G_Ksp=-R *T *log(1/Ksp);          %计算沉淀反应的标准吉布斯自由能变
delta_G_2=2 *delta_G_Ksp+delta_G_1;     %计算反应(2)的标准吉布斯自由能变
K2=exp(delta_G_2/(-R *T))
```

运算结果是：

问题（1）的平衡常数为 K1=9.5944e-007。

问题（2）的平衡常数为 K2=6.6628e+005

运算结果用 MATLAB 的科学计数法表示了，按照习惯的科学计数法：

$K1=9.5944\times10^{-7}$，$K2=6.6628\times10^{5}$

本章将在这里结束，与 MATLAB 的庞大体系相比较，本章叙述的内容实在是九牛一毛。但本章叙述了 MATLAB 的最基本内容，最基本的通常是最重要的。MATLAB 是一个工具，应该在使用过程中熟悉它，掌握它。希望读者不要把它作为一种纯理论来学，以知道了为满足，而应当以应用为目的，做到灵活地、得心应手地在实践中不断用 MATLAB 解决遇到的作图、化学计算与数据处理问题。



将数据转变为图形是极其令人愉快的，因为数据是比较抽象的，而图形则比较直观，能立即给人更多，更明确的信息。依据数据作图，也是一种重要的数据处理方法，但以前限于条件，化学工作者只能做很简单的图形，对于在化学教学与科研中遇到的较为复杂的函数或实验数据，无法将其转变为图形，从而丢掉一些能够从图形中得到的有用信息，这不能不说是一个遗憾。而 MATLAB 出现，弥补了这一缺陷。MATLAB 具有强大的作图、图像分析与处理功能，能够轻易地做出各种复杂函数的曲线、曲面图形和其它图形，使得化学工作者在教学和科研中如虎添翼。当读者看到复杂的函数或数据转变为美丽的、赏心悦目的图形，必定感到极其愉快与舒服。在本章，我们讲述如何利用 MATLAB 做函数曲线和曲面图和基于实验数据点做折线图、饼图和柱形图。

2.1　曲线图

利用 MATLAB 做一元函数曲线图的基本函数是 plot。plot 的作用是将给定横坐标和纵坐标的一系列数据点画在屏幕上，并用线段将这些数据点连起来。

使用 plot 函数，最紧要的是给 plot 提供数据点的横坐标和纵坐标，使得 plot 能够根据数据点的横坐标和纵坐标画出数据点，需要注意的是 plot 函数接受的数据点集的横、纵坐标需要分别用向量表示。此外，还需要给 plot 提供线型（如用实线还是用点划线）、曲线的颜色、画出的点的类型（如是用“*”表示点还是用“o”表示点等）。

我们举两个例子，说明 plot 函数的使用方法。

2.1.1　基态氢原子径向分布函数图

根据薛定谔方程，氢原子基态波函数为：

$$\psi_{1s}=\sqrt{\frac{1}{\pi a_0^3}}\mathrm{e}^{\frac{-r}{a_0}}$$

式中，a_0 是 Bohr 半径，其值为 52.9pm。

基态氢原子径向分布函数为：

$$D(r)=4\pi r^2\psi_{1s}^2=4\frac{1}{a_0}\left(\frac{r}{a_0}\right)^2(\mathrm{e}^{\frac{-r}{a_0}})^2$$

$D(r)$ 函数反映了核外半径 r 处电子出现的概率，是一个很重要的函数，以 r/a_0 为自变量，做出位于闭区间［0 9］径向分布函数图形。很明显，这是做一元函数曲线图的问题。

我们下面利用 MATLAB 做出径向分布函数的曲线图 $D(r)$，以了解核外电子出现的概率与半径 r 的关系。

具体 MATLAB 语句如下：

```
r_a0=0:0.1:9;                          %指定横坐标
a0=52.9;                               %Bohr 半径
Dr=4/a0 *r_a0.^2. *(exp(-r_a0)).^2;    %计算相应的纵坐标
plot(r_a0,Dr, '- *b')                  %作图,同时指定其它要素
xlabel('核外半径')                     %指定横轴标题
ylabel('电子出现概率')                 %指定纵轴标题
title('基态氢原子径向分布函数图')      %指定整个图形的名称
```

根据上述代码做出的基态氢原子径向分布函数图见图 2-1。

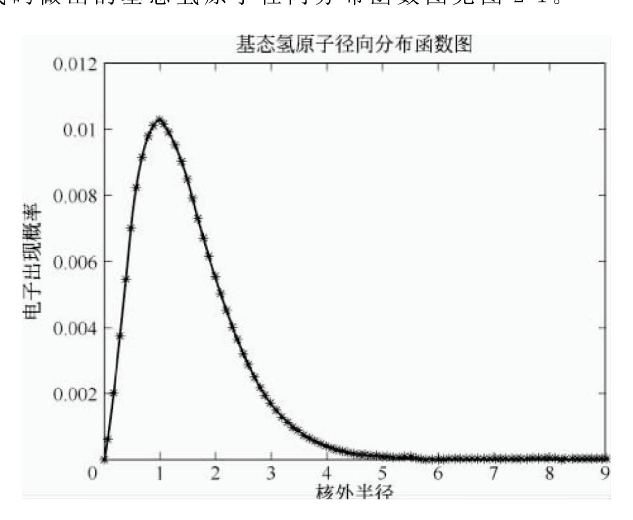


图 2-1　基态氢原子径向分布函数图

从上图可见，在 $r/a_0=0$ 处，也就是原子核处，电子出现的概率为 0，然后电子出现的概率随 r/a_0 的增加而增加，直到在 $r/a_0\approx 1$ 处达到最大值，然后随 r/a_0 的增加而减小，在 $r/a_0\approx 6$ 处，电子出现的概率基本为 0 了。如果不直接做出 $D(r)$ 的图形，上述电子出现概率与核外半径的关系很难得到，由此可以看出作图能够给化学工作者提供多么有用的信息。

上述 MATLAB 语句的第一句生成了一个向量，表示各个数据点的横坐标。第二句则依据要画的函数，通过运算，得到一个向量，获得各个数据点的纵坐标，请注意运算过程中使用了点乘算符和点乘方算符。数据点的横坐标用向量 r_a0 表

示，数据点的纵坐标用向量 Dr 表示。在获得 r_a0 和 Dr 向量后，第三句使用 plot 函数根据数据点的横坐标和纵坐标绘图。但在第三句，除了给出 r_a0 和 Dr，还指定了线型、点的类型和曲线的颜色，“-”表示用实线连接各数据点，“ * ”表示用星号表示数据点，“b”表示曲线颜色使用蓝色（blue）。关于 MATLAB 对线型、点类型和曲线颜色的定义，见表 2-1。

表 2-1　MATLAB 对线型、点类型和曲线颜色的定义

线型		曲线颜色			
-	实线	m(magenta)	品红	r(red)	红色
- -	划线	y(yellow)	黄色	g(green)	绿色
：	点线	k(black)	黑色	b(blue)	蓝色
-.	点划线	w(white)	白色	c(cyan)	青色
点类型					
＋	加号	^	向上三角形	p	正五角形
o(字母)	小圆圈	v	向下三角形	s	正方形
*	星号	＞	向右三角形	h	正六角形
.	实点	＜	向左三角形	d	菱形
×	交叉号				

xlabel，ylabel，title 都是 MATLAB 的固有函数，它们的作用是在做出图形后，指定图的横轴、纵轴和整个图形的标题，这三个函数的自变量是字符串，在 MATLAB 中，由单引号括住的字母、单词或汉字称为字符串，是一种文本信息，用于说明某些内容，如图形的名称等。用 xlabel 等函数为图形加上各种说明，虽然这不是必要的，但可以使图形清晰明了。

2.1.2　草酸各种存在形式的分布曲线

上面在一张图里只画了一条曲线，MATLAB 还允许在一张图里画出多条曲线。

从酸碱离解反应可知，当共轭酸碱对处于平衡状态时，溶液中存在着 H_3O^+ 和不同的酸碱形式。这时它们的浓度称为平衡浓度。当溶液的 pH 值发生变化时，平衡发生移动。以致酸碱存在形式的分布情况也随之发生变化。酸碱平衡体系中某种存在形式的平衡浓度占总浓度（各种存在形式的平衡浓度的总和，也称为分析浓度）的分数称为“分布系数”，以 d 表示。分布系数与 pH 值之间的关系曲线称为分布曲线。讨论分布曲线可以帮助我们深入了解酸碱滴定的过程、滴定误差以及分步滴定的可能性，而且对于了解络合滴定与沉淀反应条件等也是有用的。我们现在以二元酸草酸为例，用 MATLAB 做出草酸的存在形式的分布曲线。

草酸在溶液的存在形式是：$H_2C_2O_4$，$HC_2O_4^-$ 和 $C_2O_4^{2-}$，根据质量平衡，草酸的总浓度 c 应为以上 3 种组分存在形式的平衡浓度之和，即

$$c=[H_2C_2O_4]+[HC_2O_4^-]+[C_2O_4^{2-}]$$

如果以 d_2、d_1 和 d_0 分别代表 $H_2C_2O_4$，$HC_2O_4^-$ 和 $C_2O_4^{2-}$ 的分布系数，K_{a1} 和

K_{a2} 分别代表草酸的一级和二级离解常数，根据电离平衡关系和质量平衡，得到：

$$d_2=\frac{[H_2C_2O_4]}{c}=\frac{[H^+]^2}{[H^+]^2+K_{a1}[H^+]+K_{a1}K_{a2}}$$

$$d_1=\frac{[HC_2O_4^-]}{c}=\frac{K_{a1}[H^+]}{[H^+]^2+K_{a1}[H^+]+K_{a1}K_{a2}}$$

$$d_0=\frac{[C_2O_4^{2-}]}{c}$$

根据质量平衡，$d_2+d_1+d_0=1$。

下面我们使用 MATLAB 在一张图上将草酸的三种存在形式的分布曲线画在一张图上。横坐标用 pH 值，纵坐标是 d 值。pH 值与 $[H^+]$ 的关系是：$[H^+]=10^{-pH}$。

```
Ka1=5.9e-2;                                  %草酸一级离解常数
Ka2=6.4e-5;                                  %草酸二级离解常数
pH=0:0.1:7;                                  %指定 pH 值范围
H=10.^(-pH);                                 %由 pH 值计算氢离子浓度
delta_2=H.^2./(H.^2+Ka1*H+Ka1*Ka2);          %计算草酸的分布系数
delta_1=Ka1*H./(H.^2+Ka1*H+Ka1*Ka2);         %计算草酸一氢根的分布系数
delta_0=1-delta_2-delta_1;                   %计算草酸根的分布系数
plot(pH,delta_2,'-*b',pH,delta_1,'-or',pH,delta_0,'-hk')
                                             %作图并规定其它要素
xlabel('pH 值')                              %指定横轴标题
ylabel('分布系数')                           %指定纵轴标题
title('草酸各种存在形式的分布曲线')          %指定图形标题
legend('d2','d1','d0','Location','best')     %做出图例,区分各分布曲线
```

做出的图形见图 2-2：

在上述 MATLAB 语句中，需要注意的是在计算氢离子浓度以及计算草酸三种形式的分布系数时，使用了点乘方和点除算符，因为在上述语句里，pH 是向量，计算氢离子浓度得到的氢离子浓度也是向量，向量能够作为一个整体参加运算，是 MATLAB 的特点和优点所在，使得我们进行有关计算变得方便而简洁。在用 plot 作图的语句中，首先指定了画出草酸分布曲线所需要的数据，如草酸分布曲线的横坐标和纵坐标、曲线的线型、点的类型和曲线颜色，然后指定了画出草酸一氢根分布曲线的所需数据，最后指定画出草酸根分布曲线所需要的数据，如此，使用 plot 语句，就能在一张图上画出三条分布曲线。

最后一条语句是：legend ('d2', 'd1', 'd0', 'Location', 'best')，该语句的作用是做出图例，对各条曲线进行区分。因为一张图上画出了三条曲线，所以需要在图上指明各条分布曲线是草酸哪种形式的分布曲线，也就是需要对各分布曲线取个名字，

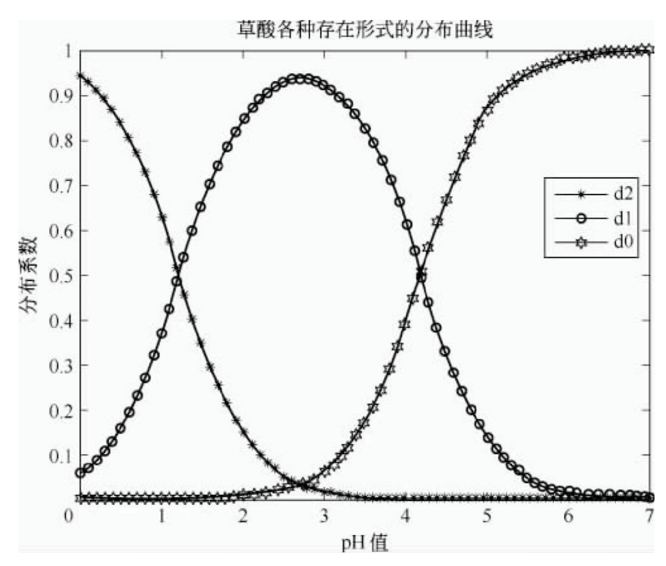


图 2-2　草酸各种存在形式的分布曲线

进行区别。我们这里依次把三种分布曲线分别命名为 d_2，d_1，d_0，然后需要将各曲线名字与曲线本身进行联系。MATLAB 可以将曲线上的点的类型与该曲线名字进行关联。在本图上，草酸分布曲线上点是星号，草酸一氢根的分布曲线上的点是圆圈，草酸根分布曲线上的点是正六角形。按照 plot 函数中做分布曲线的次序，是先做草酸分布曲线，然后做草酸一氢根的分布曲线，最后做草酸根分布曲线，我们在 legend 函数中命名的次序是 d_2，d_1，d_0，这样 d_2 就与草酸分布曲线建立了联系，d_1 就与草酸一氢根的分布曲线建立了联系，d_0 与草酸根分布曲线建立了联系。最后这种联系需要在图上表达出来，这就是图例。图例的位置用'location'结合一个表示位置的字符串来规定，我们这里采用了参数'best'，规定 MATLAB 将图例放在图上的最合适的位置，当然读者也可以根据实际情况选用'North'，'East'，'NorthEast'，'NorthWest'等参数，指定图例放在其它位置。最后请大家注意 legend 函数里的所有参数都是字符串，都加了单引号。

由分布曲线可知：当 pH<1.2 时，$H_2C_2O_4$ 占优势；当 1.2<pH<4.2，$HC_2O_4^-$ 为主要存在存在形式；当 pH>4.2 时，则 $C_2O_4^{2-}$ 占优势。我们还可以看出，$[HC_2O_4^-]$ 的分布曲线 d_1 在 pH≈2.7 时，存在一最大值，这个最大值点该如何精确求得，我们将在后面的最优化章节予以详细讲述。

2.2　折线图

化学工作者常常要根据实验数据做出折线图，以反映一个变量随另外一个变量的变化趋势或关系，如反应物浓度随反应时间的变化趋势，酸碱滴定过程溶液 pH 与碱加入的体积之间关系。做折线图基本函数仍然是 plot 函数，作图的第一步仍然

是给 plot 提供数据点的横坐标和纵坐标。我们下面通过两个例子说明用 MATLAB 做折线图的例子。

2.2.1 热解产品的产率与温度关系

农作物秸秆在缺氧条件受热会发生热分解，最终生成燃料油、炭粉和气体三种产品，这三种产品的产率受温度影响很大，不同温度下三种产品的产率，见表 2-2。试做出三种产品的产率与温度关系的折线图。

表 **2-2** 不同温度下三种热分解产品的产率（质量分类）

温度/℃	气体/%	炭粉/%	燃料油/%
420	12	32	56
450	14	25	61
480	20	24	57
510	24	19	57
540	31	17	52

做上述折线图的 MATLAB 语句如下：

```
t=[420 450 480 510 540];                                   %热分解温度
product=[12 14 20 24 31;32 25 24 19 17;56 61 57 57 52];   %三种产品产率
plot(t,product(1,:),'-*b',t,product(2,:),'-or',t,product(3,:),'-hk')   %作图
xlabel('温度℃')                                            %指定横轴标题
ylabel('热分解产品产率(质量分数)%')                          %指定纵轴标题
title('热分解产品产率与温度的关系')                          %指定图形标题
legend('气体','炭粉','燃料油','Location','best')              %做图例
```

折线图见图 2-3：

从上述的 MATLAB 语句看出，基于实验数据做折线图和做函数曲线图并无本

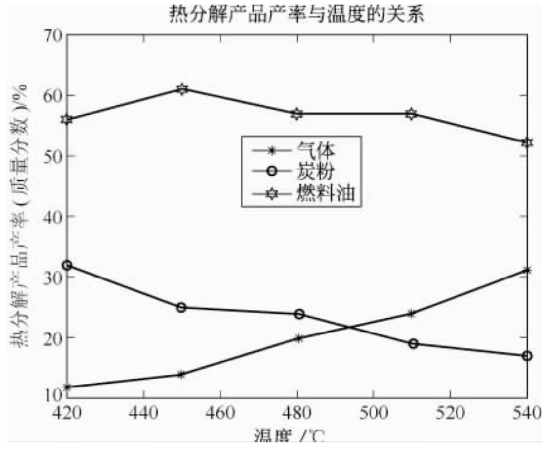


图 2-3 热分解产品产率与温度的关系

质区别，都是指定点的横坐标和纵坐标后利用 plot 函数作图。它们的区别在于做函数曲线图是根据函数所在的区间，生成等距向量指定数据点的横坐标，然后根据函数关系计算出数据点的纵坐标，进而用 plot 做出函数图形。做折线图，其数据点的横坐标和纵坐标均来自实验数据或观察数据，需要老老实实在 MATLAB 语句中输入。上述的语句的第一句定义了向量 t，储存温度数据，作为折线图的横坐标，第二句定义矩阵 product，该矩阵第一行储存气体产率，第二行定义炭粉产率，第三行定义燃料油的产率，该矩阵每行用分号（；）区别，每列用空格区别。在用 plot 作图部分，根据引用矩阵块的规则，product（1,:），product（2,:），product（3,:）分别代表矩阵 product 的第 1、2 和 3 行，分别储存气体、炭粉和燃料油的产率。

从三种产品的产率与温度关系的折线图可以看出，气体产率随着温度的升高而升高，炭粉产率随着温度的升高而降低，燃料油的产率随着温度升高先升高，达到一最大值，然后随着温度的升高，燃料油的产率降低。

2.2.2　滴定曲线

为了正确运用酸碱滴定法进行分析测定，必须了解酸碱滴定过程中 H^+ 浓度的变化规律，才有可能选择适合的指示剂，准确地确定滴定终点。以 NaOH 溶液滴定 HCl 溶液为例，随着 NaOH 溶液的不断增加，被滴定的 HCl 溶液的 H^+ 浓度不断降低，pH 值是逐渐升高的，但 pH 值的具体变化规律是怎么样的？尤其是在等当点附近 pH 值的变化规律涉及分析测定的准确程度，更加重要。这可以通过做 NaOH 溶液滴加体积 V 与被滴定溶液的 pH 值的关系曲线来确定，这曲线被称为滴定曲线。

下表列出了用 0.1000mol/L NaOH 溶液滴定 20.00mL 的 0.1000 mol/L HCl 溶液时，NaOH 溶液滴加体积 V 与被滴定溶液的 pH 值的数据，见表 2-3。要求根据表中的数据，做出滴定曲线并进行分析。

表 **2-3**　**NaOH** 溶液滴加体积 **V** 与被滴定溶液的 **pH** 值关系

NaOH 加入量/mL	0.00	18.00	19.80	19.98	20.00	20.02	20.20	22.00	40.00
pH 值	1.00	2.28	3.30	4.31	7.00	9.70	10.70	11.70	12.50

第一步仍然是给 plot 提供数据点的横坐标和纵坐标，在这个例子一共 9 个数据点，我们当然可以直接定义两个向量，一个向量存储加入 NaOH 溶液体积的数据，一个向量存储被滴定溶液的 pH 值的数据，然后作图。如果我们有 20 个，甚至 200 个数据点，这样直接定义向量会非常不方便，容易出错。在这里，我们采用文件输入方式定义向量，采用文件输入数据，在数据量特别大的情况下具有很大优越性。

我们首先用 Windows 自带的记事本编辑书写一个文本文件，这个文件的整体

格式是一矩形数据块，该矩形数据块第一列存储加入 NaOH 溶液体积的数据，第二列存储被滴定溶液的 pH 值数据。编辑的文本文件见图 2-4，我们把这个文本文件取名为 V_pH. txt，保存在 D 盘的根目录下。

```
V_pH.txt - 记事本
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)
0.00     1.00
18.00    2.28
19.80    3.30
19.98    4.31
20.00    7.00
20.02    9.70
20.20    10.70
22.00    11.70
40.00    12.5
```

图 2-4　存储 NaOH 溶液滴加体积 V 与被滴定溶液的 pH 值关系的文本文件

那么 MATLAB 如何从文件 V_pH. txt 读取这些数据，从而进行下一步的作图操作呢？MATLAB 使用 load 命令读取文本文件 V_pH. txt 的矩形块数据。具体格式如下：

```
load d:/V_pH
```

通过上述语句，MATLAB 首先自动生成一个变量，该变量的名字就是被读取的文本文件的名字，也就是说，该变量的名字就是 V_pH，接着 MATLAB 将 V_pH. txt 中矩形块数据读取出来，赋给变量 V_pH，这样，变量 V_pH 成为一个矩阵，其第一列是加入 NaOH 溶液体积的数据，第二列是被滴定溶液的 pH 值数据。

在了解了通过文件输入数据的方法后，做出滴定曲线的 MATLAB 语句如下：

```
load d:/V_pH.txt                        %从文件中读取矩阵数据，并赋给变量 V_pH
plot(V_pH(:,1),V_pH(:,2),'-*b')         %作图
xlabel(' NaOH 加入量/mL')               %指定横轴标题
ylabel('pH 值')                         %指定纵轴标题
title('滴定曲线')                       %指定图形标题
```

做出的滴定曲线见图 2-5：

通过对滴定曲线的分析得知，从滴定曲线上近似垂直的 pH 突跃部分，可以求出等当点时所需要的 NaOH 溶液体积，或者根据等当点附近的 pH 突跃，就可以选择适合指示剂。显然，在等当点附近变色的指示剂如溴百里酚蓝、苯酚红等可以正确指示滴定终点的到达，因为等当点正处于指示剂变色范围内。

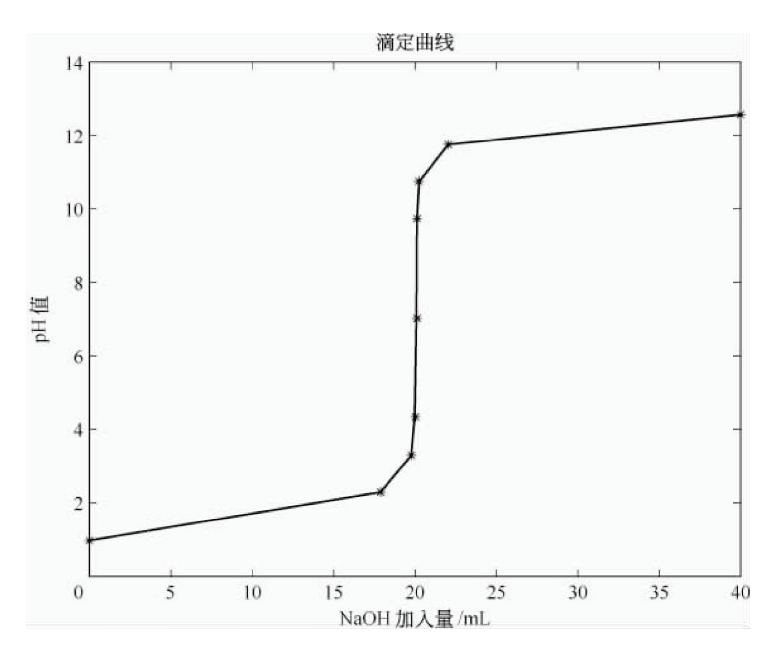


图 2-5　滴定曲线

2.3　二元函数曲面图

对于二元函数 $z=f(x,y)$，就无法使用 plot 函数对其进行作图了。但作图的基本原理是一样的。对于一元函数，用 plot 作图时，首先需要指定作图区间，即 plot 函数对一元函数作图是在一闭区间内作图。对于二元函数，闭区间概念变成了闭区域，即对二元函数作图，首先需要定义一作图的闭区域，然后利用有关函数进行作图。

我们以二元函数 $z=xe^{-x^2-y^2}$ 为例，讨论一下该如何利用 MATLAB 做出该函数的图形。

首先，我们需要定义一个作图的矩形闭区域，在这个矩形闭区域内对该函数作图。我们打算在矩形闭区域 $\{(x,y)\mid -2=x=2,-2=y=2\}$ 内对该函数作图。因此我们写下如下 MATLAB 语句，先确定该区域的范围：

```
x=-2:0.1:2;
y=-2:0.1:2;;
```

上面两条语句定义了两个向量 x 和 y，它们的分量从 -2 逐渐增大到 2，它们只是规定了这个矩形区域横向和纵向的范围，还不是这个区域本身，如果想定义这个区域本身，还需要进一步做出定义。

回想一下，我们在对一元函数作图时，所谓定义闭区间，实际上是指定该区间内某些点的坐标值，区间上点的坐标值只有一个，因为区间是一条一维的线段。同样，我们定义闭区域，实际上是指定该闭区域内某些点的坐标值，注意闭区域内点

的坐标值包括横坐标和纵坐标，因为区域是个平面，有横向和纵向两个维度，我们准备定义的闭区域内的各指定点用星号（*）表示，见图 2-6。

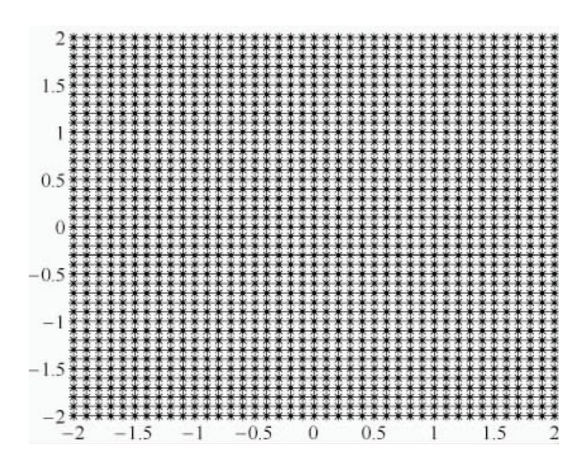


图 2-6　准备定义的闭区域内的各指定点

我们看出，这些点本身构成了一个矩形点阵，因为确定这些点就需要确定这些点的横坐标和纵坐标，所以应该有两组数据，一组数据包含这些点的横坐标，另外一组数据包含点的纵坐标，这两组数据都是以矩阵形式存在的，这两个矩阵，我们分别称为横坐标矩阵和纵坐标矩阵，这两个矩阵的维度或大小与上图的点阵一致，横坐标矩阵的每一元素就是相应点阵上对应点的横坐标值，纵坐标矩阵的每一元素就是相应点阵上对应点的纵坐标值。因此，问题就转化成如何获得横坐标矩阵和纵坐标矩阵了。MATLAB 使用 meshgrid 函数获得闭区域的横坐标矩阵和纵坐标矩阵。具体语句如下：

```
[xx,yy]=meshgrid(x,y);
```

向量 x 和 y，它们的分量从 −2 逐渐增大到 2，它们规定了这个矩形区域横向和纵向的范围，通过 meshgrid 函数，获得横坐标矩阵 xx 和纵坐标矩阵 yy。

在获得横坐标矩阵 xx 和纵坐标矩阵 yy 后，我们接着要计算闭区域点阵里的点对应的函数值，注意下列语句使用了点乘算符和点乘方算符，因为 xx 和 yy 都是矩阵，最后获得函数值矩阵 zz。语句如下：

```
zz=xx.*exp(-xx.^2 - yy.^2);
```

xx，yy，zz 是相同大小的矩阵，并且(xx(m,n),yy(m,n),yy(m,n))是我们最后要画出的曲面上的某个点的坐标，因为曲面上的点是三维的，所以该点有三个分量。注意这里的点是指曲面上的点。前面的点指的是闭区域的点，闭区域的点有两个分量。

在得到 xx，yy，zz 的矩阵后，也就是知道我们所要画的曲面上各点的三个坐标后，我们就可以把这个曲面画出来了，MATLAB 使用 mesh 函数画出曲面，语句如下：

mesh(xx,yy,zz)

我们把上述语句综合，做出二元函数 $z=xe^{-x^2-y^2}$ 曲面图的 MATLAB 语句如下：

```
x=-2:0.1:2;                            %规定矩形区域横向范围
y=-2:0.1:2;                            %规定矩形区域纵向范围
[xx,yy]=meshgrid(x,y);                 %获得横坐标矩阵 xx 和纵坐标矩阵 yy
zz=xx .*exp(-xx.^2 - yy.^2);           %获得函数值矩阵 zz
mesh(xx,yy,zz)                         %根据曲面上各点的三个坐标值作图
xlabel('\it x \rm 轴')                 %指定横轴标题
ylabel('\it y \rm  轴')                %指定纵轴标题
zlabel('\it z \rm  轴')                %指定 z 轴标题
title('二元函数 z=x *exp(-x *x-y *y)曲面图')
                                       %指定曲面标题
```

做出的曲面图形见图 2-7。

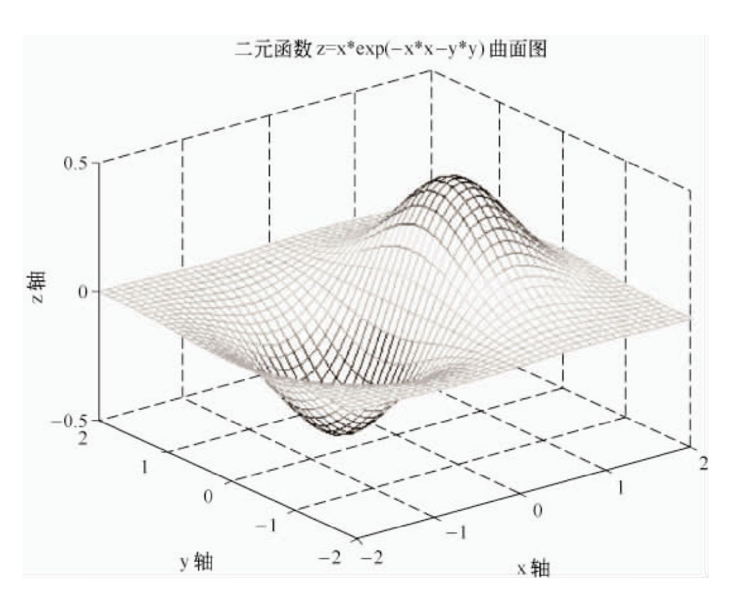


图 2-7　二元函数 $z=xe^{-x^2-y^2}$ 的曲面图

请注意在 xlable、ylable 和 zlable 函数中的自变量字符串中使用了控制符 \ it 和 \ rm。\ it 的作用是将跟在它后面的字符变成斜体字，但本身并不输出。比如'\ it x'的作用是将常规字体的 x 变为斜体 *x*。而 \ im 的作用是将跟在它后面的字符变成常规字体，但本身并不输出。MATLAB 的字符控制符很多，通过这些控制字符，可以将常规字体变成斜体、黑体、规定字符的颜色、为字符添加边框，还可以输出希腊字母和其它特殊字符（如©、♥），具体使用方法请读者参考 MATLAB 帮助文档。

2.3.1 中压条件下氮气的 $\boldsymbol{P=f(v,T)}$ 曲面图

中压条件（几十个大气压）下的气体的 P-v-T 关系，可以用范德华方程描述：

$$v=\frac{RT}{P+\frac{a}{v^2}}+b$$

已知氮气的范德华常数 $a=1.351\times10^6\text{atm}\cdot\text{cm}^3/\text{mol}$，$b=38.64\ \text{cm}^3/\text{mol}$，

试做出中压条件下氮气的 $P=f(v,T)$ 曲面图。

我们知道，气体分子密度越大（摩尔体积越小），温度越高，气体压力越大，这对任何气体都是适用的，是压力与气体的摩尔体积与温度定性关系。但我们现在要把中压条件下氮气压力与其摩尔体积和温度的关系通过一个曲面图表达出来，在作图前，我们当然需要把上面的范德华方程改写为：

$$P=\frac{RT}{v-b}-\frac{a}{v^2}$$

做出中压条件下氮气的 $P=f(v,T)$ 曲面图的 MATLAB 语句如下：

```
a=1.351e6;                                   %氮气的范德华常数 a
b=38.64;                                     %氮气的范德华常数 b
R=82.06;                                     %采用大气压作为压力单位的气体常数
v=500:1000;                                  %规定矩形区域横向范围
T=(-150+273.15):1:(100+273.15);              %规定矩形区域纵向范围
[vv,TT]=meshgrid(v,T);                       %获得横坐标矩阵 xx 和纵坐标矩阵 yy
PP=R * TT./(vv-b)-a./vv.^2;                  %获得函数值矩阵 zz
mesh(vv,TT,PP)                               %根据曲面上各点的三个坐标值作图
xlabel('摩尔体积/(毫升/mol)')                %指定横轴标题
ylabel('温度/K')                             %指定纵轴标题
zlabel('压力/atm')                           %指定 z 轴标题
title('中压条件下氮气的 P=f(v,T)曲面图')     %指定曲面标题
```

中压条件下氮气的 $P=f(v,T)$ 的曲面图见图 2-8：

从中压条件下氮气的 $P=f(v,T)$ 曲面图可以看出，确实是温度越高，气体的摩尔体积越小，气体的压力越大。但从曲面图可以明显看出，温度对气体的压力影响很大，而气体的摩尔体积对压力的影响较小。

2.3.2 水烃比和总压对乙苯转化率的影响

乙苯脱氢制取苯乙烯：

$$C_6H_5-C_2H_5 \longrightarrow C_6H_5-C_2H_3+H_2$$

是一分子数增加的可逆吸热反应。为降低组分分压，维持反应温度，工业反应

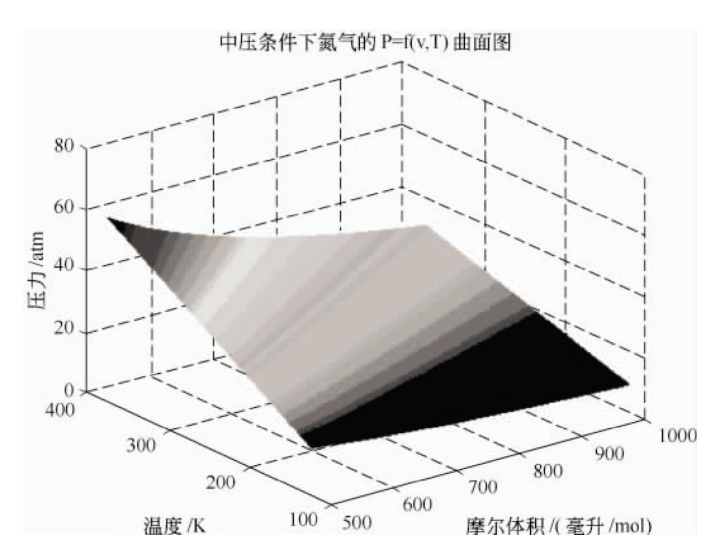


图 2-8　中压条件下氮气的 $P=f(v,T)$ 曲面图

器中用大量水蒸气对反应物系进行稀释。进料中水蒸气和乙苯的质量比称为水烃比（SOR），是反应器的一个重要工艺参数。又由反应动力学研究可知，反应温度过高会导致苯乙烯选择性下降。因此反应在 525℃ 下进行，此温度下反应平衡常数 $K_p=0.4539\times10^{-2}$ MPa，试研究该温度下水烃比和总压（绝压）对乙苯转化率的影响。

根据反应平衡计算，乙苯的转化率 X 与水烃比 SOR 和总压（绝压）p_t 的关系是：

$$X=\frac{-\alpha+\sqrt{\alpha^2+4\left(\frac{p_t}{K_p}+1\right)(\alpha+1)}}{2\left(\frac{p_t}{K_p}+1\right)}$$

$\alpha=106\times\text{SOR}/18$，$\alpha$ 是进料中水蒸气和乙苯的摩尔比。

根据以上关系，我们做出二元曲面图 $X=f(\text{SOR},p_t)$，并分析水烃比和总压（绝压）对乙苯转化率的影响。

自然，首先需要定义函数 $X=f(\text{SOR},p_t)$，定义函数 $X=f(\text{SOR},p_t)$的 MATLAB 代码如下：

```
function x=f(SOR,pt)
Kp=0.4539e-2;                 %525℃下反应的平衡常数,单位 MPa
alpha=106 *SOR/18;            %进料中水蒸气和乙苯的摩尔比
x=(-alpha+sqrt(alpha.^2+4 *(pt/Kp+1). *(alpha+1)))./(2 *(pt/Kp+1));
                              %计算 x
```

我们设水烃比的范围是 [0.5 2.5]，总压的范围是 [0.05 0.2]。

然后书写 MATLAB 代码做出二元曲面图 $X=f(\text{SOR},p_t)$：

```
SOR=0.5:0.1:2.5;                          %规定矩形区域横向范围,水烃比
pt=0.05:0.1:0.2;                          %规定矩形区域纵向范围,总压(绝压)
[SOR_x,pt_y]=meshgrid(SOR,pt);            %获得横坐标矩阵和纵坐标矩阵
x=f(SOR_x,pt_y);                          %获得函数值矩阵,乙苯转化率
mesh(SOR_x,pt_y,x)                        %根据曲面上各点的三个坐标值作图
xlabel('水烃比')                           %指定横轴标题
ylabel('总压(绝压)/Mpa')                   %指定纵轴标题
zlabel('乙苯转化率')                        %指定z轴标题
```

水烃比和总压对乙苯转化率的影响见图2-9。从图2-9可以看出，乙苯的平衡转化率随总压降低，随水烃比的提高而提高，特别是总压对平衡转化率的影响更为敏感。但反应中水烃比不能过大，水烃比过大将导致能耗过大。因此采用负压操作有可能在较小水烃比下达到较高的转化率。以上分析就是20世纪70年代以来广泛采用的负压法乙苯脱氢的热力学基础。

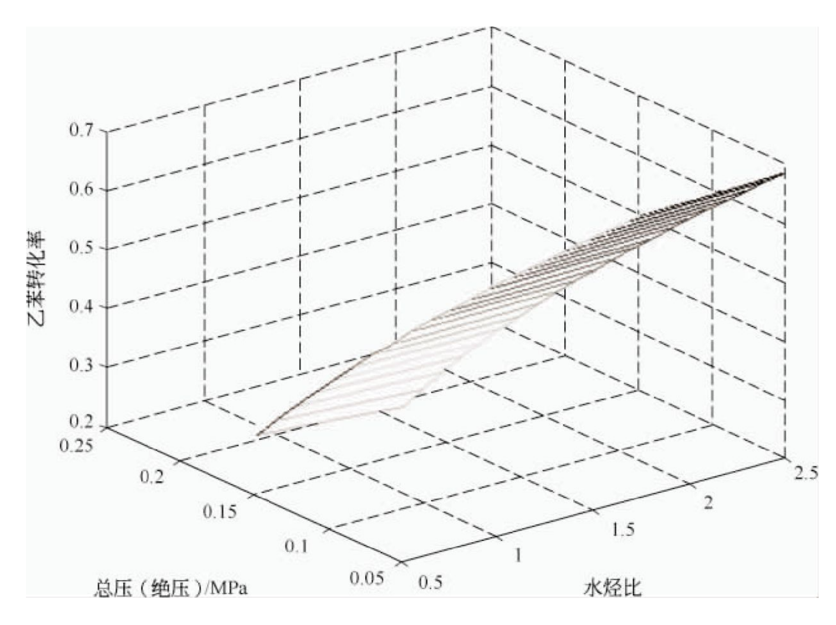


图2-9　水烃比和总压对乙苯转化率的影响

2.4 隐函数作图

隐函数指的这样一类函数 y 虽然是 x 的函数，但不能明确表达为 $y=f(x)$ 的形式，只能用方程 $g(x,y)=0$ 这样一种形式表达，现在，我们就要谈一谈如何做出一元隐函数 $g(x,y)=0$ 的函数图形。

假设有函数 $z=g(x,y)$，我们已经知道这是一个曲面，我们可以按照前面给出的方法做出这个曲面。如果我们用平面 $z=0$ 去截这个曲面，那么平面 $z=0$ 与曲面 $z=g(x,y)$ 的交线就是函数 $g(x,y)=0$ 的图形。从数学上或地理上说，交线 $g(x,y)=0$ 也被称为曲面 $z=g(x,y)$ 的高度为 0 的等高线。现在的问题就转变为在做出曲面 $z=g(x,y)$ 的图形后，如何做该曲面高度为 0 的等高线，如果该等高线能够做出，那么也就做出了隐函数 $g(x,y)=0$ 的图形。

MATLAB 使用 contour 函数做曲面的等高线，格式为：

contour (x,y,z,[v v])

其中 x，y 是用 meshgrid 函数生成的，关于函数 $z=g(x,y)$ 的横坐标矩阵和纵坐标矩阵，z 是由函数关系 $z=g(x,y)$ 生成的函数值矩阵。[v v] 代表用画出 $g(x,y)=v$ 的等高线。

我们以函数 $y^3+\sin(x)-2x-e^y=0$ 为例子，看一下这个函数该如何作图。MATLAB 语句如下：

```
x=-4:0.1:6;                                    %规定矩形区域横向范围
y=-4:0.1:6;                                    %规定矩形区域纵向范围
[xx,yy]=meshgrid(x,y);                         %获得横、纵坐标矩阵 xx、yy
zz=yy.^3+sin(xx)-2*xx-exp(yy);                 %获得函数值矩阵 zz
contour(xx,yy,zz,[0 0],'b')                    %做 zz=0 时的蓝色等高线
xlabel('\it x \rm 轴')                          %指定横轴标题
ylabel('\it y \rm 轴')                          %指定纵轴标题
title('隐函数 y*y*y+sin(x)-2*x-exp(y)=0 的图形')
                                               %指定曲线标题
```

图形见图 2-10：

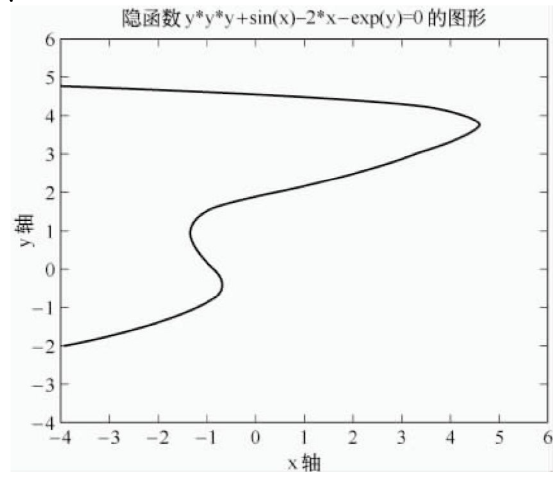


图 2-10　隐函数 y*y*y+sin(x)-2*x-exp(y)=0 的图形

上述作隐函数图的前四句和前面讲述的做 $z=f(x,y)$ 曲面的语句是一样的，实际上隐函数作图就是要先做出 $z=f(x,y)$ 曲面，然后再做出该曲面的高度为 0 的等高线，该等高线就是我们需要的隐函数图形。在得到矩阵 xx，yy，zz 后，我们实际上已经得到了曲面 $z=f(x,y)$ 的作图数据，然后我们依据这些数据，利用做等高线的函数 contour，画出等高线。需要注意的是，做高度为 0 的等高线，指定高度时用的是一长度为 2 的零向量 [0 0]，而不是单个 0。

我们接下来再来考虑如何做出二元隐函数 $f(x,y,z)=0$ 的图形，很明显，该函数图形应该是一个曲面。那么该如何画出这个曲面呢？能不能遵循上面画等高线的思路画出这个曲面呢？答案肯定的。虽然函数 $w=f(x,y,z)$ 的图形是无法画出来的，它是四维空间的一个几何体，但如果该几何体被四维空间的一个平面所截，获得的交集部分则是三维空间的一张曲面。三维空间的曲面是可以画出来的。因此，答案就出来了，做二元隐函数 $f(x,y,z)=0$ 的图形，就是做出四维空间超几何体 $w=f(x,y,z)$ 在 $w=0$ 处的等值面，注意，这里就不是等高线了，而是等值面。MATLAB 中做等值面的函数是 isosurface（xx，yy，zz，w，0）。xx，yy，zz，w 是 3 个坐标矩阵和函数值矩阵，这 4 个矩阵，给出了四维空间超几何体上的点，把这些点连起来，就能“画”出这个超几何体。但事实上，我们不画这个超几何体，而是用 $w=0$ 去截这个超几何体，获得二元隐函数曲面图。

我们以二元隐函数 $x^2+y^2+z+\sin(z)-1.5=0$ 为例，看如何做出该隐函数曲面图：

```
x=-4:0.1:4;                          %规定立方体区域横向范围
y=-4:0.1:4;                          %规定立方体区域纵向范围
z=-4:0.1:4;                          %规定立方体区域竖向范围
[xx,yy,zz]=meshgrid(x,y,z);          %获得横、纵和竖坐标矩阵 xx、yy、zz
w=xx.^2+yy.^2+zz+sin(zz)-1.5;        %获得函数值矩阵 zz
isosurface(xx,yy,zz,w,0)             %根据 xx,yy,zz,w4 做出 w=0 时的等值面
xlabel('\it x \rm  轴')               %指定横轴标题
ylabel('\it y \rm  轴')               %指定纵轴标题
zlabel('\it z \rm  轴')               %指定竖轴标题
title('隐函数 x *x+y *y+z+sin(z)-1.5=0 的图形')
                                     %指定曲面标题
```

做出的图形见图 2-11。

2.4.1 中压条件下氮气的 $v=f(P,T)$ 曲面图

我们再来考虑描述中压条件下氮气范德华方程：

$$v=\frac{RT}{P+\frac{a}{v^2}}+b$$

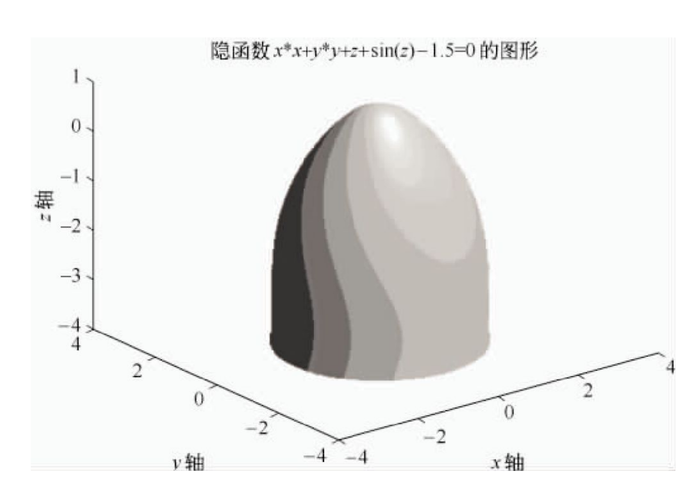


图 2-11　隐函数 $x*x+y*y+z+\sin(z)-1.5=0$ 的图形

式中，$a=1.351\times10^6\text{atm}\cdot\text{cm}^3/\text{mol}$，$b=38.64\text{cm}^3/\text{mol}$。

这次我们想做出 $v=f(P,T)$ 的图像，考察中压条件下氮气的摩尔体积是如何被温度和压力影响的，从范德华方程可以看出，摩尔体积 v 是温度 T 和压力 P 的二元隐函数，我们用二元隐函数作图的方法，做出 $v=f(P,T)$ 曲面。

```
a=1.351e6;                                  %氮气的范德华常数 a
b=38.64;                                    %氮气的范德华常数 b
R=82.06;                                    %采用大气压作为压力单位的气体常数
P=20:5:150;                                 %规定矩形区域横向范围
T=(-150+273.15):10:(150+273.15);
v=10:10:800;                                %规定矩形区域纵向范围
[PP,TT,vv]=meshgrid(P,T,v);                 %获得横坐标矩阵 xx 和纵坐标矩阵 yy
w=R *TT./(PP+a./vv.^2)+b-vv;                %获得函数值矩阵 zz
isosurface (PP,TT,vv,w,0)                   %根据曲面上各点的三个坐标值作图
xlabel('压力/atm')                           %指定横轴标题
ylabel('温度/K')                             %指定纵轴标题
zlabel('摩尔体积/(毫升/mol)')                 %指定 z 轴标题
title('中压条件下氮气的 v=f(P,T)曲面图')       %指定曲面标题
```

做出的曲面见图 2-12，从曲面图可知，中压条件下氮气的摩尔体积随温度的升高而显著升高，随着压力减小具有一定的升高。

2.4.2　不同过程膨胀功的比较

在 298K 时，有一定量的单原子理想气体（$C_{V,m}=1.5R$），从始态 2000kPa 及

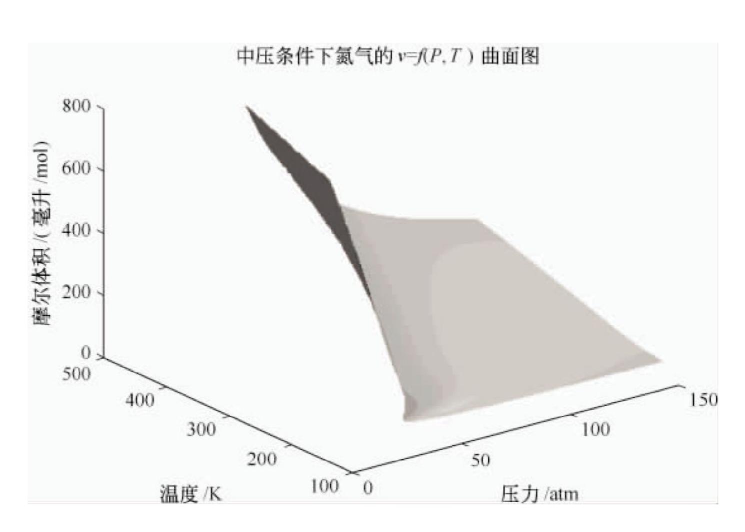


图 2-12 中压条件下氮气的 $v=f(P,T)$ 曲面图

$20dm^3$ 经过下列不同过程膨胀到终态压力为100kPa。

① 等温可逆膨胀

② 绝热可逆膨胀

③ 以 $n=1.3$ 的多方过程膨胀

试在 P-V 图上画出三种膨胀功的示意图，并比较三种功的大小。

对等温可逆膨胀：$PV=2000*20$

由于单原子理想气体，$\gamma=C_{p,m}/C_{V,m}=(1.5R+R)/1.5R=5/3$

对绝热可逆膨胀：$PV^{5/3}=2000*20^{5/3}$

以 $n=1.3$ 的多方过程膨胀：$PV^{1.3}=2000*20^{1.3}$

我们利用隐函数做图法做出这三种过程的 P-V 图，具体的MATLAB代码如下：

```
p=100:5:2000;                              %规定矩形区域纵向范围
v=20:10:2000*20/100;                       %规定矩形区域横向范围
[vv,pp]=meshgrid(v,p);                     %获得横、纵坐标矩阵xx、yy
zz=pp.*vv-2000*20;                         %获得函数值矩阵zz
contour(vv,pp,zz,[0 0])                    %做等温线
text(2000*20/600,600,'\leftarrow 等温过程','FontSize',9)
                                           %标注等温线
hold on
v=20:10:(2000*20^(5/3)/100)^(3/5);         %规定矩形区域横向范围
[vv,pp]=meshgrid(v,p);                     %获得横、纵坐标矩阵xx、yy
zz=pp.*vv.^(5/3)-2000*20^(5/3);            %获得函数值矩阵zz
contour(vv,pp,zz,[0 0])                    %做绝热线
```

```
text((2000 * 20^(5/3)/400)^(3/5),400,'\leftarrow  绝热过程','FontSize',9)
                                              %标注绝热线
hold on
v=20:10:(2000 * 20^(1.3)/100)^(1/1.3);        %规定矩形区域横向范围
[vv,pp]=meshgrid(v,p);                        %获得横、纵坐标矩阵 xx、yy
zz=pp. * vv.^1.3-2000 * 20^(1.3);             %获得函数值矩阵 zz
contour(vv,pp,zz,[0 0])                       %做多方过程线
text((2000 * 20^1.3/200)^(1/1.3),200,'\leftarrow  多方过程','FontSize',9)
                                              %标注多方过程线
xlabel('\it V\rm /dm^3')                      %指定横轴标题
ylabel('\it P\rm /kPa')                       %指定纵轴标题
hold off
```

做出的图形见图 2-13：

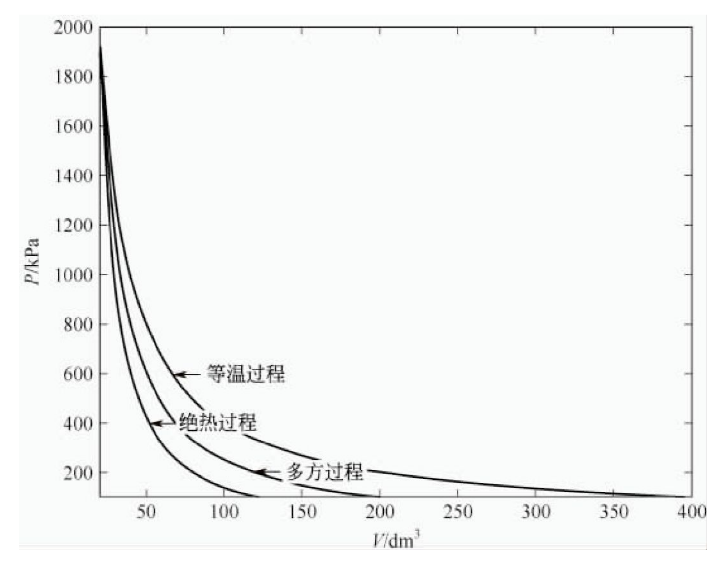


图 2-13　不同过程膨胀功的比较

上述代码用到了 hold on 命令，其作用是将下一次画的图，画在上一次画图的图形窗口上，即把前后两次画的图，画在同一个窗口上。上述代码用了两次 hold on 命令，就可以将三条过程曲线画在同一个图形窗口里，从而利于比较。在代码末尾使用了 hold off 命令，其作用是终止 hold on 命令，使用了 hold off 命令后，下一次画的图将画在新开的图形窗口上。

上述代码使用了 text 函数，其作用是在图形的指定位置书写指定的文字。对于 text 通常的使用格式，我们举例如下：

text(2000 * 20/600,600,'\leftarrow 等温过程','FontSize',9)

其作用是在坐标[2000 * 20/600，600]处，书写字符串'← 等温过程'，注意，在MATLAB中，用\leftarrow代表字符串'←'，'FontSize'，9代表书写的字符串的大小采用9磅大小。

过程做的功应该是过程曲线与纵轴（P 轴）所围的面积。从图2-12看出，等温可逆膨胀过程做功最多，多方可逆膨胀过程做功次之，绝热可逆膨胀做功最小。

2.5 饼图和柱形图

饼图可以表示一个整体怎样分成几个部分。要画饼图，先要画个圆，圆代表全体。圆里面的扇形代表各个部分，各扇形的圆心角和各部分的大小成比例。饼图的好处是让我们看到所有部分合起来，确实是全体，总体的某一部分占全体多大比例。但由于角度比长度难比较，所以做饼图不是比较各部分大小的好方法。

MATLAB使用pie函数做饼图，pie函数的基本使用格式是pie(x)，x 是个向量，其分量就是所要表达的各个部分的大小，MATLAB根据各部分的大小自动计算各部分占整体多大的比例。

2.5.1 我国2002年常规能源构成

我国2002年消费的常规能源构成见表2-4：

表**2-4** 我国**2002**年消费的常规能源构成

占能源生产总量的比重/%			
原煤	原油	天然气	水电
70.7	17.2	3.2	8.9

试做出反映上述表格数据的饼图。

MATLAB语句如下：

```
pie([70.7 17.2 3.2 8.9])        %做饼图
legend('原煤','原油','天然气','水电','Location','southeast')
title('我国2002年常规能源构成')
```

做出的饼图见图2-14：

从图上可以看出，我国的能源构成中，煤的消费占了绝大部分，由此造成很多问题，如燃煤产生的二氧化硫污染问题，采煤造成的地面塌陷问题等，对原油的消费也占了我国能源消费的相当部分，造成我国目前对外石油依存度超过30%，因此，我国目前迫切需要改变能源消费结构，大力发展水电、风电和生物质能等可再生能源。

从图中可以注意到，饼图和图例稍有重叠，可以在MATLAB的图形窗口用鼠标一直点住图例，按着不动，然后进行拖动，将图例拖动到更适合的位置。

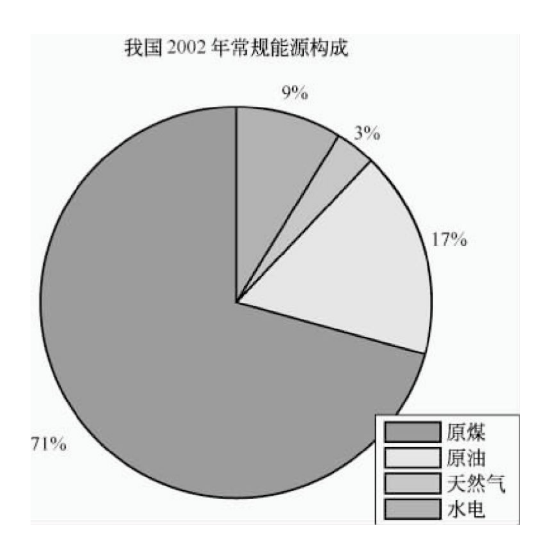


图 2-14　我国 2002 年常规能源构成

柱形图将一个整体各部分的大小表示为具有一定高度的矩形，由于比较高度比比较角度容易，因此，柱形图是比较各部分大小的好方法。此外，柱形图还可以比较并不属于同一整体的数量，而这是饼图所不能做到的。

2.5.2　地壳中分布最广的 5 种元素的原子含量

地壳中分布最广的 5 种元素的原子含量见表 2-5。

表 **2-5**　地壳中分布最广的 **5** 种元素的原子含量

O	H	Si	Al	Na
52.32%	16.95%	16.67%	5.53%	1.95%

试用 MATLAB 做出反映这 5 种元素原子含量的柱形图。

做出该柱形图的 MATLAB 语句如下：

```
bar([0.5232 0.1695 0.1667 0.0553 0.0195;zeros(1,5)])        %做柱形图
legend('O','H','Si','Al','Na','Location','northeast')
title('地壳中分布最广的 5 种元素的原子含量')
```

得到柱形图见图 2-15。

从上述柱形图直观的看出，对于这 5 种原子，地壳中氧原子的含量最高且远高于其它原子，氢原子的含量硅原子的含量基本相等，钠原子的含量最低。

上述语句的第一句，使用了 MATLAB 的固有作图函数 bar，bar 函数的作用就是根据提供的数据做出柱形图。bar 函数的基本使用格式是 pie([x;zeros(1,length(x))]，x 是个行向量，其分量就是所要表达的各个部分的大小，我们前面提过，

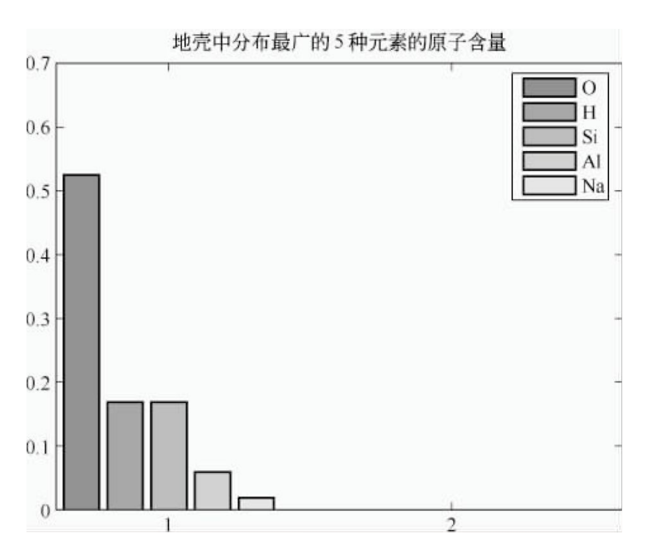


图 2-15　地壳中分布最广的 5 种元素的原子含量

zeros 是 MATLAB 的固有函数，其作用是产生一全部元素为零的矩阵，在这里，zeros (1，length(x))产生一长度与行向量 x 长度相等的行向量，length(x)也是 MATLAB 的固有函数，其作用是返回向量 x 的长度，即返回向量 x 有多少分量。请注意 [x;zeros(1,length(x))]其实进行了矩阵合并，即把行向量 x 和全零行向量 zeros(1,length(x))合并为一个行数为 2，列数为 length(x)的矩阵，最后用这个矩阵作为 bar 函数的自变量。虽然这样做有点奇怪，但这是做出上面柱形图的必须步骤。

依据作者的实践，认为用 MATLAB 做饼图和柱形图并不方便，而且做出的图形也不够令人满意，作者推荐读者使用 Excel 做饼图和柱形图，使用 Excel，能够方便的做出令人满意的饼图和柱形图。

2.6　MATLAB 的图形格式

由 MATLAB 做出的图形后，可以将做出图形在 MATLAB 的窗口保存为扩展名为 fig 的图形文件或扩展名为 bmp、JPG 格式的图形文件，fig 格式的图形文件是 MATLAB 的专有图形文件，MATLAB 可以直接读取编辑。扩展名为 bmp、JPG 格式是目前图形的标准格式，可以被当前流行的各种图形处理软件编辑处理，也可以插入各种文档中，本章中 MATLAB 做出的图形，就是由 MATLAB 做出后，保存为 bmp 文件，然后插入本章中的。bmp 格式的图形文件包含较多的像素，因此比较清晰，建议读者在保存 MATLAB 图形时，采用 bmp 格式进行保存。

MATLAB 的图形窗口见图 2-16。

在 MATLAB 的图形窗口，点 File 菜单中的 save 选项或 save as 选项，可以将 MATLAB 的做出的图形保存为 fig、bmp、JPG 等格式，并可以选择保存位置。

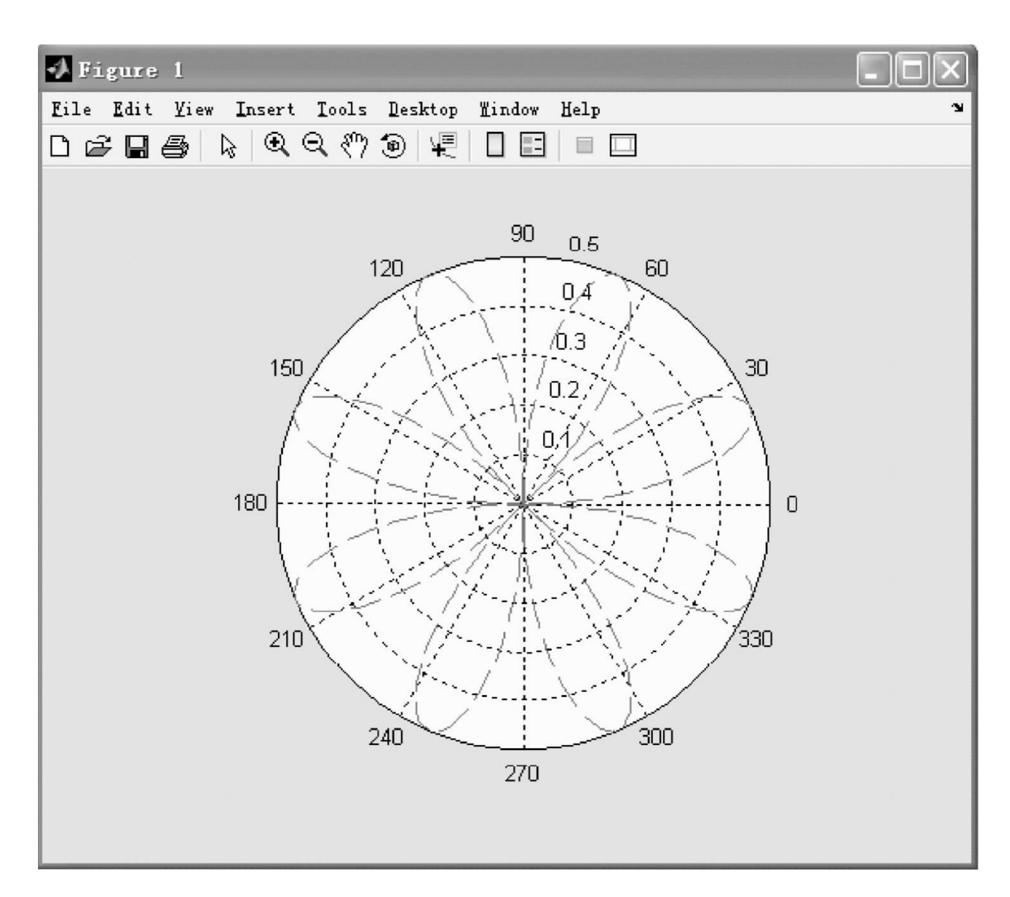


图 2-16 MATLAB 的图形窗口

最后，MATLAB 允许读者做出图形后，在 MATLAB 自身的图形窗口对图形或 fig 格式的图形文件做出各种各样的修改，如在图中添加文本，去掉坐标轴，移动图的位置等各种操作，通过这些操作，读者能够做出更符合自身要求的图形；此外，MATLAB 还可做出极坐标图、双对数图、三维曲线图、彗星图等各种图形，有关这方面的内容，请读者查看 MATLAB 的联机帮助文档或在线帮助。



3.1 非线性方程的求解

在日常的数学计算中，我们实际上只会解一元一次方程、一元二次方程和能够进行因式分解的一元高次方程，对于一般的非线性方程，我们则束手无策，现在，我们就要来看一看，对于一般的非线性方程，该如何求解。

3.1.1 不动点迭代法与维格斯坦（Wegstein）加速

将非线性方程 $f(x)=0$ 改写为 $x=g(x)$，如果存在 $x^*=g(x^*)$，我们就称 x^* 是函数 $g(x)$ 的不动点，自然，x^* 也是方程 $f(x)=0$ 的根。由此引出不动点迭代方法：

① 将方程 $f(x)=0$ 改写为 $x=g(x)$；

② 大致估计出不动点 x^* 的范围，给 x^* 设定一个近似值 x_0 以进行不动点迭代。如果不能估计出不动点 x^* 的范围，则不动点迭代的初值 x_0 可以任意取；

③ 将初值 x_0 代入函数 $g(x)$，得 $x_1=g(x_0)$；

④ 将获得的 x_1 作为新的初值再代入函数 $g(x)$，得 $x_2=g(x_1)$；

⑤ 如此反复将代入函数 $g(x)$ 获得的值又作为下一次的初值代入 $g(x)$，从而获得一个序列 $[x_0\ x_1\ x_2\cdots x_n\ x_{n+1}\cdots]$，称为不动点迭代；

⑥ 直到 $|g(x_{n+1})-x_{n+1}|\leqslant\varepsilon$ 为止（ε 是一个给定的很小的正数），此时认为 x_{n+1} 就是函数 $g(x)$ 的不动点，也是方程 $f(x)=0$ 的根。如果迭代到 m 次仍然达不到此要求（m 是一个很大的正整数），则认为不动点迭代方法是无效的，此时称迭代是发散的或不收敛的。

不动点迭代方法的优点在于简单方便。但不动点迭代方法对初值要求比较高，即迭代初值 x_0 一般要比较接近不动点 x^*，迭代方能收敛。此外，$f(x)=0$ 可以改写为多种形式的 $x=g(x)$，$g(x)$ 的形式对迭代是否收敛也有很大影响。由于上述两个缺点，直接使用不动点迭代求非线性方程的根并不是好的方法。为了克服不动点迭代的缺点，保留不动点迭代方法的优点的，人们提出了各种迭代加速方法，其中最具代表性的就是维格斯坦（Wegstein）加速方法：

① 将方程 $f(x)=0$ 改写为 $x=g(x)$；

② 大致估计出不动点 x^* 的范围，给 x^* 设定一个近似值 x_0 以进行不动点迭代。

如果不能估计出不动点 x^* 的范围，则不动点迭代的初值 x_0 可以任意取；

③ 将初值 x_0 代入函数 $g(x)$，得 $x_1=g(x_0)$；

④ 在以后的迭代中采用以下关联式：

$x_{p+1}=tg(x_p)+(1-t)x_p$

其中 $t=\frac{1}{1-s}=\frac{1}{1-\frac{g(x_p)-g(x_{p-1})}{x_p-x_{p-1}}}, p=1,2,\cdots$；

⑤ 获得迭代序列$[x_0\ x_1\ x_2\cdots x_n\ x_{n+1}\cdots]$；

⑥ 直到$|g(x_{n+1})-x_{n+1}|\leqslant\varepsilon$为止（ε是一个给定的很小的正数），此时认为 x_{n+1} 就是函数 $g(x)$的不动点，也是方程 $f(x)=0$ 的根。如果迭代到 m 次仍然达不到此要求（m 是一个很大的正整数），则认为维格斯坦加速方法是无效的，此时称迭代是发散的或不收敛的。

维格斯坦加速方法与不动点迭代方法的区别就在于第 4 步，即产生 $x_2\cdots x_n$ $x_{n+1}\cdots$的方式。经过实践证明，维格斯坦加速方法是比不动点迭代方法更优越的方法，能够有效求解非线性方程 $f(x)=0$。

该法实际上是一种加权迭代方法，其权值 t 由最近二次估算的结果来确定。事实上，不同的权值 t 代表了不同含意。

$t=1$ 表示不动点迭代。

$t<1$ 表示慢速稳健的迭代，可以有效避免函数分散或震荡。

$t>1$ 表示快速迭代，但函数迭代的稳定性变差，容易发散或震荡。

尽管权值 t 可以为任意一个实数，但经过精密的数学推理，对于化学化工的计算过程，以 $1\leqslant t\leqslant 6$ 作为权值 t 的取值范围已被证明是最有利和可靠的。

我们当然可以按照上面给出步骤编写一段程序代码，从而实现维格斯坦加速方法，求解非线性方程的根。但更好的办法是我们把这段代码编写为一个 MATLAB 函数，保存起来，这个函数文件既可以被直接使用，求任意非线性方程的根，又可以被其它函数或代码调用，作为实现某个特定功能的一个子模块，从而大大增强程序代码的通用性与方便性。实现维格斯坦（Wegstein）加速方法的 MATLAB 函数文件如下。

```
function root=Wegstein_root(g_name,x0,tolerance,m)
%     维格斯坦(Wegstein)加速方法求非线性方程的根,如迭代不收敛,返回“无穷大”inf
%     g_name:字符串或函数句柄,函数名,也是函数文件名,定义函数 g(x)
%     x0 : 迭代初值
%     tolerance : 允许误差,可选择的参数,默认值 1e-6
%     m:最大迭代次数,默认值 200
n=nargin;                          %确定函数自变量个数
if n<3
```

```
    tolerance=1e-6;                     %默认迭代精度
end
if n<4
    m=200;                              %默认迭代次数
end
x1=feval(g_name,x0);                    %计算 g(x0)
k=0;
if      abs(x1-x0)>tolerance            %设定迭代条件 1
        while 1
            s=(feval(g_name,x1)-feval(g_name,x0))/(x1-x0);
            t=1/(1-s);                  %以上语句确定权值 t
            x2=t *feval(g_name,x1)+(1-t)*x1;
                                        %产生新的迭代初值
            if abs(feval(g_name,x2)-x2)<tolerance
                                        %设定迭代条件 2
                break                   %如符合迭代条件 2,跳出循环
            else
                x0=x1;
                x1=x2;
            end
            k=k+1;
            if k>m
                root=inf;
                return
            end
        end;
else
        root=(x0+x1)/2;                 %符合迭代条件 1,返回方程的根
        return
end
root=x2;                                %符合迭代条件 2,返回方程的根
```

nargin是MATLAB的一个固有函数，但此函数不带自变量，返回值是所在函数的实际输入自变量的个数。nargin函数对于输入自变量个数不确定的函数特别有用。如上面求非线性方程根的Wegstein_root函数，需要有迭代精度和最大迭代次数。用户可以输入自己规定的精度和最大迭代次数，也可以不输入这两个数据，采用Wegstein_root函数内定的默认值。要做到这一点，就需要启用nargin函数，判断Wegstein_root函数有几个输入值，如果函数的输入值少于4个或3个，就表明用户

没有规定最大迭代次数或迭代精度，从而利用选择语句，采用函数内定的默认值。

feval 函数也是 MATLAB 的固有函数，其使用格式是 feval（函数名，函数自变量），函数名是一代表函数文件名的字符串常数或函数句柄，用于指定待求值函数，其作用是求被函数名指定的函数在相应自变量处的函数值。举例如下：

```
≫sin(1)
ans=0.8415
≫feval('sin',1)
ans=0.8415
```

我们可以看出，feval（'sin'，1）的作用和 sin（1）作用是完全一样的，都是求 1 的正弦值，但为什么会有 feval 函数存在呢？是不是多此一举呢？答案是否定的。拿 Wegstein_root 函数为例，它的作用是求任意非线性方程的根，因此对应的迭代函数 $g(x)$ 也具有任意性，我们事先无法知道 $g(x)$ 有什么样的形式，会取什么函数名，这一切，都需要根据问题具体定义。我们能够做到的是在定义函数 $g(x)$ 后，把相应的函数名字以字符常数的形式，作为 Wegstein_root 函数的自变量，传递给 Wegstein_root 函数，以便 Wegstein_root 函数能够根据获得的函数名进行求根操作。求根操作的关键步骤之一就是求函数 $g(x)$ 的值，这里，就用到 feval 函数了，feval 函数根据 $g(x)$ 的函数名对 $g(x)$ 函数求值。

while…end 循环语句的循环条件是 1，表明这个循环入口处不进行判断，直接进入循环，因为我们必须先执行一次求权值 t 和产生新迭代值的操作，然后进行判断。我们把判断操作让循环体内的 if 语句去执行，如果满足收敛条件，则跳出 while 循环。break 命令是 MATLAB 的固有命令，作用是跳出 break 当前所在的循环，执行循环后面的操作。

请注意，Wegstein_root 函数里设定的迭代条件 1 和迭代条件 2 本质上是一样的，都是规定当 $|g(x_{n+1})-x_{n+1}|=\varepsilon$ 时（ε 是一个给定的很小的正数），认为迭代收敛，求根成功。只是由于维格斯坦（Wegstein）加速方法产生迭代序列时，第一步用的是不动点迭代，后面则通过加权的方式产生迭代序列值。

return 函数的作用是退出本函数，返回上一级调用本函数的函数或调用本函数的一段代码。return 常常用在选择结构语句中，如果达到一定条件，则执行 return 操作。在 Wegstein_root 函数中，如果迭代次数超过 m，则认为迭代发散或不收敛，则非线性方程的根定义为 inf，接着退出 Wegstein_root 函数，返回上一级调用本函数的函数或调用本函数的一段代码。

3.1.1.1　中压条件下氮气的摩尔体积

我们来看一个用 Wegstein_root 函数求非线性方程根的例子：

中压条件（几十个大气压）下的气体的 P-V-T 关系，可以用范德华方程描述：

$$v=\frac{RT}{P+\frac{a}{v^2}}+b$$

已知氮气的范德华常数 $a=1.351\times10^6$ atm·cm^3/mol，$b=38.64$ cm^3/mol，确定温度为－100℃，压力为50atm（1atm＝101325Pa）下的氮气的摩尔体积。

根据题意，上述非线性方程已经具有 $x=g(x)$ 的形式，因此 $g(x)$ 函数定义如下：

```
function y=g(v)
R=82.06;T=273.15+(-100);P=50;
a=1.351e6;b=38.64;
y=R * T/(P+a/v^2)+b;
```

MATLAB允许把多条语句写在同一行，只要各条语句通过分号（;）相隔。再次需要提醒的是，所有定义的函数在编写完成后，一般要保存在MATLAB安装目录下的work子目录中。

我们利用理想气体状态方程来产生求解该范德华方程的初值，求解温度为－100℃，压力为50atm下的氮气的摩尔体积的MATLAB语句如下：

```
R=82.06;T=273.15+(-100);P=50;
v0=R * T/P;
vm=Wegstein_root('g',v0)
```

运行这段代码，得到：

vm＝222.4455

温度为－100℃，压力为50atm下的氮气的摩尔体积是222.4455cm^3/mol。

3.1.1.2 REDLICH-KWONG状态方程

REDLICH-KWONG状态方程是比范德华在更大温度、压力范围内有效的气体状态方程，被广泛应用于工程计算并享有盛誉。

REDLICH-KWONG状态方程是：

$$v=\frac{RT}{P}-\frac{a}{pT^{0.5}v}+b$$

其中 P 为压力，T 是温度，v 是摩尔体积，R 为气体常数，a、b 是与具体气体有关的特性常数，由下两式给出：

$$a=\frac{0.4278R^2T_c^{2.5}}{P_c} \qquad b=\frac{0.0867RT_c}{P_c}$$

T_c 和 P_c 分别是气体的临界温度和临界压力。

当温度是热力学温度，压力是大气压时，$R=82.05$，气体的摩尔体积的单位是cm^3/mol。

试用REDLICH-KWONG状态方程计算异丙醇蒸气在10atm和473K时的摩尔体积。已知异丙醇的临界温度是508.2K，临界压力是50atm。

因为REDLICH-KWONG状态方程就是 $x=g(x)$ 的形式，因此可以直接采用

维格斯坦（Wegstein）加速方法，定义函数 g(v)，如下：

```
function y=g(v)
    Tc=508.2;
    pc=50;
    R=82.05;
    a=0.4278 *R^2 *Tc^2.5/pc;
    b=0.0867 *R *Tc/pc;
    p=10;
    T=473;
    y=R *T/p-a/(p *T^0.5 *v)+b;
```

我们利用理想气体状态方程来产生求解的初值，计算异丙醇蒸气在 10atm 和 473K 时的摩尔体积的 MATLAB 语句如下：

```
R=82.05;T=473;P=10;
v0=R *T/P;
vm=Wegstein_root('g',v0)
vm=3.5145e+003
```

因此，异丙醇蒸气在 10atm 和 473K 时的摩尔体积是 $3514.5cm^3/mol$。

3.1.2　对分法

尽管维格斯坦（Wegstein）加速方法已经足够好，但仍然有失效的可能。我们这里将要讨论另外一种方法，只要非线性方程 $f(x)=0$ 确实有根，那么该方法一定能将 $f(x)=0$ 的根求出来。这种方法就是对分法。

根据高等数学中零点定理，如果函数 $f(x)$ 在闭区间 $[a, b]$ 连续，并且 $f(a)*f(b)<0$，那么必然在闭区间 $[a, b]$ 内存在一点 x^*，使得函数 $f(x^*)=0$。

对分法就是根据闭区间连续函数的零点定理衍生出来的方法，对分法求根原理示意图见图 3-1：

（1）首先确定，对于闭区间 $[a, b]$，$f(a)*f(b)<0$，这表明非线性方程的根 x^* 在闭区间 $[a, b]$ 内，闭区间 $[a, b]$ 被称为有根区间。一般这一步可以根据实际问题，做函数 $f(x)$ 的曲线图来确定；

（2）在确认非线性方程的根 x^* 在闭区间 $[a, b]$ 内之后，用区间中点 $c=(a+b)/2$ 分割闭区间 $[a, b]$；

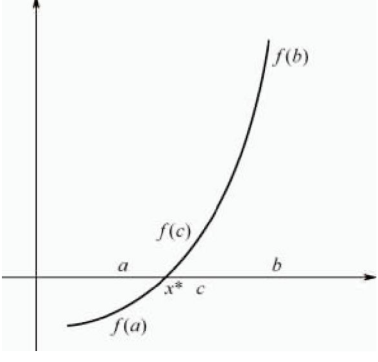


图 3-1　对分法求根原理

(3) 如果 $f(a)*f(c)<0$，则表明根 $x*$ 在闭区间 $[a, c]$ 内，这时，有根区间缩小为闭区间 $[a, c]$ 内。否则，根 $x*$ 在闭区间 $[c, b]$ 内，有根区间缩小为闭区间 $[c, b]$；无论如何，有根区间的长度缩小为原来长度的一半。

(4) 重复步骤2和3，直到有根区间长度小于 ε 为止（ε 是一个给定的很小的正数），取有根区间的中点作为非线性方程 $f(x)=0$ 的根。有时在进行对分操作时，区间中点的函数值的绝对值非常接近0，即 $|f(c)|=\varepsilon$（ε 是一个给定的很小的正数），这时，也可以停止对分，取区间中点 c 为非线性方程的根。

对分法的效率很高，每对分一次，就能将有根区间长度缩短一半，对分 n 次得到的有根区间长度就变为原始有根区间长度的 2^{-n}，并且，只要知道了非线性方程有根区间，就一定能通过有限次对分获得足够精度的解。因此，对分法是求非线性方程根的有效方法。

实现对分法的MATLAB函数文件如下：

```
function root=bisec_n(f_name,a,b,tolerance)
%        对分法求非线性方程的根,如无根,返回"无穷大"inf
%        f_name:字符串或函数句柄,函数名,定义待求根的方程
%        a,b : 有根区间的下限和上限,建议采用作图方法确定有根区间
%        tolerance : 允许误差,可选择的参数,默认值 1e-6
n=nargin;                    %确定函数自变量个数
if n<4
   tolerance=1e-6;     %默认最小有根区间长度
end
Y_a=feval(f_name,a); Y_b=feval(f_name,b);
if ( sign(Y_a * Y_b)==1)
      root=inf;              %如区间内无根,返回 inf
else
   while 1
      c=(a+b)/2;
      if abs(feval(f_name,c))<1e-10
         root=c;          %如区间中点的函数值足够小,返回区间中点作为根
         return
      end
      if abs(b-a)<=tolerance
         root=c;          %如有根区间长度足够小,返回区间中点作为根
         return
      end
```

```
        Y_c=feval(f_name,c);      %以下语句进行对分操作
        if( sign(Y_a * Y_c)==-1)
            b=c;
        else
            a=c;Y_a=Y_c;
        end
    end
end
```

bisec_n 函数中使用了 sign 函数，sign 函数是 MATLAB 的固有函数，其作用是根据数值的正负号，返回相应的数值。如果数值为正，返回正数 1，如果数值为 0，返回 0，如果数值为负，返回-1。

3.1.2.1 螯合物溶液 pH 值的改变

室温下，已知 0.010mol/L EDTA-2Na（Na_2H_2Y）溶液的 pH 值为 4.46，在此溶液中加入 $Cu(NO_3)_2$，使得 $c(Cu^{2+})$ 为 0.010 mol/L（设体积不变），当生成螯合物 $[CuY]^{2-}$ 的反应达到平衡后，问溶液的 pH 值改变了多少？

本例中发生如下的螯合反应，该反应的平衡常数是 200。

$$Cu^{2+} + H_2Y^{2-} = [CuY]^{2-} + 2H^+$$

本问题实际是一个化学平衡计算问题，反应体系的化学平衡计算可看成一类特殊的物料衡算问题，即体系的终了状态是化学平衡状态的物料衡算问题。

设该反应的反应进度是 ksai，得

	Cu^{2+}	+	H_2Y^{2-} =	$[CuY]^{2-}$	+	$2H^+$
达到平衡时	0.010-ksai		0.010-ksai	ksai		$2*ksai+10^{-4.46}$

根据化学平衡关系：

$$ksai*(2*ksai+10^{-4.46})^2-200*(0.010-ksai)*(0.010-ksai)=0$$

得到螯合平衡后，溶液的 pH 值改变应该是 $-\lg(2*ksai+10^{-4.46})-4.46$，因此，如果能从上面的方程中求解出 ksai，那么 pH 的改变也就迎刃而解了。

我们使用对分法求解平衡方程中的 ksai，为此，需要先编写定义平衡方程的函数文件，注意，定义函数时采用了点乘和点乘方算符，这是为了下面作图的方便。

```
function y=f(ksai)
y=ksai. *(2 *ksai+10^(-4.46)).^2-200 *(0.010-ksai). *( 0.010-ksai);
```

为了判断有根区间，我们做出函数 f(ksai) 的图像，语句如下：

```
ksai=0:0.0001:0.01;
y=f(ksai);
plot(ksai,y)
```

做出的 f(ksai) 函数图像见图3-2：

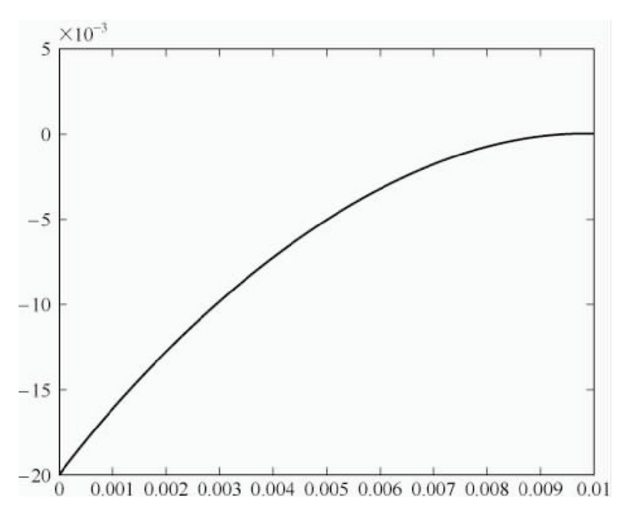


图3-2 函数 f(ksai) 的图像

从 f(ksai) 的图像可知，非线性方程 f(ksai)=0 确实在闭区间［0，0.01］有一根，因此编写如下MATLAB代码：

```
ksai=bisec_n('f',0,0.01)
delta_pH=-log10(2*ksai+10^(-4.46))-4.46
```

运行上述代码，得到：

ksai=0.0099

delta_pH=-2.7557

说明加入 $Cu(NO_3)_2$ 后，溶液的pH值降低了2.7557。

本题亦可使用维格斯坦加速方法求解，为此，编写 $g(x)$ 函数如下：

```
function y=g(ksai)
y=ksai+ksai.*(2*ksai+10^(-4.46)).^2-200*(0.010-ksai).*(0.010-ksai);
```

根据化学计量学常识，该螯合反应的反应进度ksai应该介于0到0.01中间，因此我们取迭代初值为0.005。求解过程的MATLAB语句如下：

```
ksai=Wegstein_root('g',0.005)
delta_pH=-log10(2*ksai+10^(-4.46))-4.46
```

运行上述代码：

ksai=0.0099

delta_pH=-2.7553

从 delta _pH 的结果可以看出，利用维格斯坦加速方法和对分法求得结果具有细微差别，这是由于两者求根的方式是不一样的，但两者求得的根都在给定的精度内，虽然具有细微差别，但都是合理而正确的结果。

我们这里再用对分法求解前面用维格斯坦加速方法求解过的温度为－100℃，压力为 50atm 下的氮气的摩尔体积的问题。

首先需要编写函数文件：

```
function y=f(v)
R=82.06;T=273.15+(-100);P=50;
a=1.351e6;b=38.64;
y=v-R *T./(P+a./v.^2)-b;
```

注意，上述函数文件使用了点除和点乘方算符，是为了后续作图方便。

上述函数的作图语句如下：

```
v=1:1000;
y=f(v);
plot(v,y)
```

函数 $f(v)$ 的图形见图 3-3，可见，闭区间［1 1000］是非线性方程 $f(v)=0$ 的有根区间，因此求 $f(v)=0$ 的对分法的 MATLAB 代码如下：

```
bisec_n('f',1,1000)
```

运行，得到

```
ans=222.4455
```

利用对分法同样求得，温度为－100℃，压力为 50atm 下的氮气的摩尔体积是 222.4455 cm^3/mol。

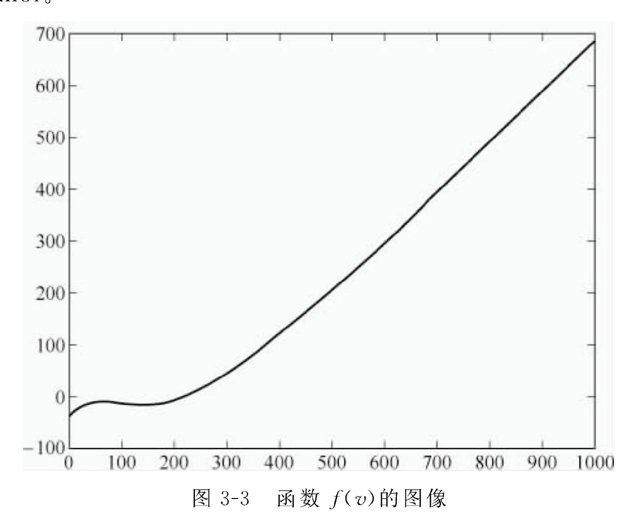


图 3-3　函数 $f(v)$ 的图像

3.1.2.2 水的热分解

在容积为 $v=10.0\mathrm{L}$ 的容器内放有 1.00mol 水，当加热到 $T=1750\mathrm{K}$ 时，发生如下反应：

$$2H_2O \longrightarrow 2H_2 + O_2$$

其平衡常数 $K^{\ominus}=1.89\times10^{-9}$（此处标准压力 $p^{\ominus}$ 按 1atm 计），计算平衡时氧气的物质的量。

设达到平衡时氧的物质的量是 x 摩尔，则根据上述反应方程式，氢气的物质的量是 $2x$ 摩尔，水蒸气的物质的量是 $1-2x$ 摩尔，根据平衡关系，得到：

$$K^{\ominus}=\frac{[p(H_2)/p^{\ominus}]^2[p(O_2)/p^{\ominus}]}{[p(H_2O)/p^{\ominus}]^2}$$

根据理想气体方程，得到各气体的分压：

$p(O_2)=x*R*T/v$

$p(H_2)=2*x*R*T/v$

$p(H_2O)=(1-2x)*R*T/v$

将各分压代入平衡方程，我们将会得到一个关于 x 的三次分式方程，我们采用对分法求解，首先定义函数文件 f，该文件定义了关于 x 的函数 $f(x)$，我们要用对分法求函数 $f(x)$的零点，即

$$f(x)=\frac{[p(H_2)/p^{\ominus}]^2[p(O_2)/p^{\ominus}]}{[p(H_2O)/p^{\ominus}]^2}-k^{\ominus}$$

具体定义如下：

```
function y=f(x)
k=1.89e-9;                                                  %平衡常数
R=8.314;                                                    %理想气体常数
T=1750;                                                     %温度
v=10/1000;                                                  %容器体积
p_O2=x*R*T/v;                                               %氧气分压
p_H2=2*x*R*T/v;                                             %氢气分压
p_H2O=(1-2*x)*R*T/v;                                        %水蒸气分压
p_biao=101325;                                              %标准态
y=(p_O2/p_biao).*(p_H2/p_biao).^2./(p_H2O/p_biao).^2-k;    %平衡方程
```

由于水分解的平衡常数很小，我们可以预计氧气的平衡摩尔数很小，因此我们计算 $f(0)$和 $f(0.1)$的值，看 $f(x)$零点是否在［0 0.1］区间内。运行 MATLAB，得到：

```
>>f(0)
ans=-1.8900e-009
```

≫f(0.1)

ans＝0.0897

$f(0)$与$f(0.1)$异号，说明$f(x)$的零点确实在［0 0.1］区间内，即达到平衡时氧气的摩尔数在［0 0.1］区间内，利用 bisec_n 函数求解：

x＝bisec_n('f',0,0.1)

解得：

x＝3.1738e－004

所以，达到水分解平衡时，氧气的物质的量是3.2×10^{-4}摩尔。

3.1.3 roots 函数

非线性方程$f(x)=0$中有一类特殊方程，就是一元n次方程，对于这类方程，虽然可以使用前面介绍的维格斯坦（Wegstein）加速方法和对分法求解，但 MATLAB 具有专门的 roots 函数来求解一元n次方程。

对于一元n次方程：

$a_nx^n+a_{n-1}x^{n-1}+\cdots a_2x^2+a_1x+a_0=0$

roots 函数的求解格式是：

roots([a_n a_{n-1} … a_2 a_1 a_0])

roots 函数接受的自变量是一向量，该向量的分量依次是该一元n次方程从高次项系数依次到低次项的系数，直到常数项。roots 函数以向量形式返回该一元n次方程所有的n个根。

3.1.3.1 一元弱酸溶液的 pH 值

一元弱酸 HA 的水溶液，有下列质子转移反应：

$$HA \rightleftharpoons H^+ + A^-$$

$$H_2O \rightleftharpoons H^+ + OH^-$$

若已知 HA 的分析浓度c、HA 的电离常数K_a和水的离子积K_w，求该弱酸溶液的 pH 值。

根据 HA 的电离平衡和水的离解平衡，可得如下的一元三次方程，具体推导过程可参见分析化学教材的有关章节。

$$[H^+]^3+K_a[H^+]^2-(cK_a+K_w)[H^+]-K_aK_w=0$$

我们以 0.20 mol/L 的一氯乙酸溶液为例，看如何使用 roots 函数求该一氯乙酸溶液的 pH 值，$pK_a=2.86$。

MATLAB 代码如下：

```
c＝0.20;
Ka＝10^(－2.86);
Kw＝1e－14;
r＝roots([1 ,Ka ,－(c * Ka＋Kw) ,－Ka * Kw]);
pH＝－log10(r)
```

运行，得到

pH＝1.7615－1.3644i

　　1.7975

　　13.3010－1.3644i

因为 roots 函数返回该一元 3 次方程的 3 个根，所以求 pH 值时必然返回 3 个值，但因为三个根中只有一个是合理的，因此 pH 值只有一个是合理的，从结果看出，3 个 pH 值只有一个是实数值，其它两个是虚数值，不合理，所以 0.20mol/L 的一氯乙酸溶液的 pH 值是 1.7975。

3.1.3.2　SO_2 氧化

已知 $2SO_{2(g)}+O_{2(g)}=\!=\!=2SO_{3(g)}$，在 1062K 的平衡常数 $K^{\ominus}=0.955$（此处标准压力 $p^{\ominus}$按 1atm 计）。如在该温度下，某容器含 SO_2、O_2 和 SO_3 三种气体，其分压分别为 30.4kPa、60.8kPa 和 25.33kPa。如果反应在恒温恒容下进行，求达到平衡时各物质的分压是多少？

根据题意，反应开始时：

$p(SO_2)/p^{\ominus}=0.3$，$p(O_2)/p^{\ominus}=0.6$，$p(SO_3)/p^{\ominus}=0.25$

设达到平衡时，氧气的分压减少了 Δp，设 $x=\Delta p/p^{\ominus}$。

由于反应在恒温恒容下进行：

$2SO_{2(g)}$　　＋　　$O_{2(g)}$　　$=\!=\!=$　　$2SO_{3(g)}$

$0.3-2x$　　　　$0.6-x$　　　　$0.25+2x$

得到

$$K^{\ominus}=\frac{(0.25+2x)^2}{(0.3-2x)^2(0.6-x)}=0.955$$

上述方程可以化为：

$$3.82x^3+0.56x^2+1.78x+0.011=0$$

这是一元三次方程，可以用 roots 函数求解：

r＝roots([3.82 0.56 1.78 0.011])

解得：

r＝－0.0702＋0.6784i

　－0.0702－0.6784i

　－0.0062

复数根是不合理的，因此有 $x=-0.0062$，x 是负值说明氧气的分压实际是增加的。因此 SO_2、O_2 和 SO_3 三种气体的分压可以计算如下：

```
x=-0.0062;
p_biao=101.325;                    %标准压力
p_SO2=(0.3-2*x)* p_biao            %SO2 分压,单位 kPa
p_O2=(0.6-x)* p_biao               %O2 分压,单位 kPa
p_SO3=(0.25+2*x)* p_biao           %SO3 分压,单位 kPa
```

解得：

p _SO2＝31.6539

p _O2＝61.4232

p _SO3＝24.0748

3.2　常微分方程的初值问题

知道某一物理量（如物质的浓度、电压或距离等物理量）在初始时刻的值，要求计算以后任意时刻该物理量的值的问题，经常可以归结以下形式的常微分方程：

$$\frac{dy}{dx}=f(x,y)$$

$$y(x_0)=y_0$$

这就是常微分方程的初值问题。

我们先讨论如何求解常微分方程的初值问题，然后再转到常微分方程组的初值问题。常微分方程的初值问题通常通过离散化的方法求解。所谓离散化方法，就是我们不追求获得解析式 $y(x)$，而追求对于任意一点 x_n，我们根据给出的常微分方程，建立一种递推关系，从 $y(x_0)=y_0$ 出发，逐步计算出 $y_n=y(x_n)$。

对上述常微分方程两边取定积分，积分区间是 $[x_{n-1}\ x_n]$，设积分区间的长度是 h_n。

$$\int_{x_{n-1}}^{x_n} y'dx=\int_{x_{n-1}}^{x_n} f(x,y)dx$$

上述积分式左边是很容易积分的，但右边就不容易积分了。如果积分区间的长度 h_n 足够小，那么我们就能用矩形的面积来近似右边的积分，见图 3-4：

因此，我们得到：

$$y_n-y_{n-1}=f(x_{n-1},y_{n-1})*h_n$$

整理，得：

$$y_n=f(x_{n-1},y_{n-1})*h_n+y_{n-1}$$

由于我们知道 $y(x_0)=y_0$，因此只要选择合适的 h_n，就可以通过上式，经过递推计算，得到 y_n。每步递推计算的 h_n，称为步长。这种计算方法被称为欧拉方法。

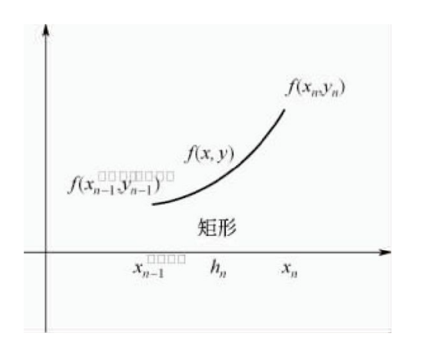


图 3-4　用矩形面积近似 $f(x,y)$ 在区间 $[x_{n-1},\ x_n]$ 的积分

很明显，上述计算方法的精度取决于能够以多大精度计算出函数 $f(x,y)$ 的定积分，欧拉方法实际上是用一条水平直线去近似函数 $f(x,y)$ 来计算定积分的，如果能够采用更高精度的方法去近似计算 $f(x,y)$ 的定积分，那么递推计算的精度还能显著提高。此外，如果能够采用极小的步长，计算精度也能提高，但采用极小的步长，可能耗费大量的计算时间，得不偿失，因此，在每次递推计算选取步长时，应当选择一合适步长，既达到了要求的计算精度，又不过度消

耗计算时间，综合这两方面的考虑，就引入了嵌入式龙格-库塔方法。

嵌入式龙格-库塔方法采用更高精度的方法计算 $f(x,y)$的定积分值，在每次递推计算选择步长时，总选择能够符合计算精度的最大步长或接近最大步长，显著降低了计算量，节省了计算时间和内存。

嵌入式龙格-库塔方法的递推计算格式如下：

```
y1=y0+h *(c(1)*K1+c(2)*K2+c(3)*K3+c(4)*K4+c(5)*K5+c(6)*K6)
```

h 是本次递推计算的步长，其它各项的意义如下：

```
K1=f(x0,y0)
K2=f(x0+a(2)*h,y0+b(2,1)*K1 *h);
K3=f(x0+a(3)*h,y0+b(3,1)*K1 *h+b(3,2)*K2 *h);
K4=(x0+a(4)*h,y0+b(4,1)*K1 *h+b(4,2)*K2 *h+b(4,3)*K3 *h);
K5=f(x0+a(5)*h,y0+b(5,1)*K1 *h+b(5,2)*K2 *h+b(5,3)*K3 *h+
    b(5,4)*K4 *h);
K6=f(x0+a(6)*h,y0+b(6,1)*K1 *h+b(6,2)*K2 *h+b(6,3)*K3 *h+
    b(6,4)*K4 *h+b(6,5)*K5 *h)
a=[0 1/5 3/10 3/5 1 7/8];
b=[0 0 0 0 0;1/5 0 0 0 0;3/40 9/40 0 0 0;3/10 -9/10 6/5 0 0;-11/54 5/2
  -70/27 35/27 0;1631/55296 175/512 575/13824 44275/110592 253/4096 ];
c=[2825/27648 0 18575/48384 13525/55296 277/14336 1/4];
d=[37/378 0 250/621 125/594 0 512/1771];
```

a、b、c、d 都按照 MATLAB 格式写成了向量或矩阵形式。

按照上面给出的递推格式，只要能够确定当次递推格式的步长，就能够逐步计算，最终求的我们需要的给定点的函数值。

那么如何确定步长呢？确定本次递推的步长是否合适，关键是估计出本次递推计算得到的结果和真值的误差 err，其次要设定一个允许误差 tolerance，即本次递推计算得到的结果和真值之间的允许的差值，一般将 tolerance 设定为 1e−5。

如果 err<tolerance，证明本次递推计算的步长合理，否则，本次递推计算选取的步长不合理，需要选择修正的当次递推计算步长。

$$\text{修正的当次递推计算步长}=\left(\frac{\text{tolerance}}{\text{err}}\right)^{0.2}*h$$

为了安全起见，根据 err 和 tolerance 得到的修正的当次递推计算步长应当适当缩小，一般取修正的当次递推计算步长乘以 0.9 做为修正的当次递推计算步长。

奇妙的是，如果本次递推步长合理，由 err 和 tolerance 得到的修正的当次递推计算步长并不是没有用的，该数值可以估计下次递推计算的步长，下次递推计算的

步长的估计值为修正的当次递推计算步长乘以 0.8。

那么 err 如何估计呢？估计算式如下：

err=max(abs(h *(cd(1)*K1+cd(2)*K2+cd(3)*K3+cd(4)*K4+cd(5)*K5+cd(6)*K6)))

cd=c-d

上述估计误差 err 的计算式子也遵循 MATLAB 格式。

若 $dy/dx=f(x,y)$，$y(a)=y0, a<b$

求 $y(b)$的嵌入式龙格-库塔方法的 MATLAB 函数如下，主函数 Rk_ad，主函数 Rk_ad 调用了子函数 RKEx：

```
function [x,y]=Rk_ad(a,b,y0,f_name,tolerance)
    % 嵌入式龙格-库塔方法
    % a,b:求解区间
    % y0:初值,数或向量
    % f_name:M 文件名,定义 f(x,y)
    % tolerance:误差限,可选择参数,默认值 1e-5
    % x:列向量,第一个分量是 a,最后一个分量是 b,其余是介于(a,b)的数值
    % y:列向量或矩阵,矩阵的每列代表与列向量 x 相对应的函数值
    m=nargin;
    if m<5
        tolerance=1e-5;
    end
    if size(y0,1)==1                    %保证 y0 是列向量
       y0=y0';
    end
    x=[a];y=[y0];
    h=(b-a)/20;                         %设定初始步长
    x0=a;
  while 1
    [z ,err]=RKEx(f_name,x0,y0,h);      %得到本次递推计算结果和误差 err
    if err<eps                          %不让 err 过小,以防止下次步长过大
        err=tolerance *1e-3;
    end
    h0=(tolerance/err)^0.2 *h;          %修正的本次递推计算步长
    if err<tolerance
        x=[x x0+h];y=[y z];
```

```
            x0=x(length(x));y0=y(:,size(y,2));
            h=h0 *0.8;                      %估计下次递推计算的步长
        else
            h=h0 *0.9;                      %重新进行本次递推计算的步长
        end
        if x0+h>=b                          %设定整个递推计算停止条件
            break
        end
      end
        z=RKEx(f_name,x0,y0,b-x0);          %进行区间端点处的计算
        x=[x b];y=[y z];
        x=x';y=y';
    function [z ,err]=RKEx(f_name,x,y0,h)
    %本函数返回本次递推计算结果 z 和误差 err
         if size(y0,1)==1
            y0=y0';
         end
         a=[0 1/5 3/10 3/5 1 7/8];
         b=[0 0 0 0 0;1/5 0 0 0 0;3/40 9/40 0 0 0;3/10 -9/10 6/5 0 0;-11/54
 5/2 -70/27 35/27 0;1631/55296 175/512 575/13824 44275/110592 253/4096 ];
         c=[2825/27648 0 18575/48384 13525/55296 277/14336 1/4];
         d=[37/378 0 250/621 125/594 0 512/1771];
         K1=feval(f_name,x,y0);
         if size(K1,1)==1
            K1=K1';
         end
         K2=feval(f_name,x+a(2)*h,y0+b(2,1)*K1 *h);
         if size(K2,1)==1
            K2=K2';
         end
         K3=feval(f_name,x+a(3)*h,y0+b(3,1)*K1 *h+b(3,2)*K2 *h);
         if size(K3,1)==1
            K3=K3';
         end
         K4=feval(f_name,x+a(4)*h,y0+b(4,1)*K1 *h+b(4,2)*K2 *h+
 b(4,3)*K3 *h);
```

```
        if size(K4,1)==1
            K4=K4';
        end
        K5=feval(f_name,x+a(5)*h,y0+b(5,1)*K1 *h+b(5,2)*K2 *h+
              b(5,3)*K3 *h+b(5,4)*K4 *h);
        if size(K5,1)==1
              K5=K5';
        end
        K6=feval(f_name,x+a(6)*h,y0+b(6,1)*K1 *h+b(6,2)*K2 *h+
              b(6,3)*K3 *h+b(6,4)*K4 *h+b(6,5)*K5 *h);
        if size(K6,1)==1
              K6=K6';
        end
        z=y0+h *(c(1)*K1+c(2)*K2+c(3)*K3+c(4)*K4+c(5)*K5+
            c(6)*K6);
        cd=c-d;
        err=max(abs(h *(cd(1)*K1+cd(2)*K2+cd(3)*K3+cd(4)*K4+
              cd(5)*K5+cd(6)*K6)));
    end
```

将上述两函数分别以文件形式保存在MATLAB的安装目录的work子目录下，以后就可以使用了。Rk_ad函数中用到了MATLAB固有常数eps，eps称为机器精度，约为2^{-52}。

下面我们举两个包含常微分方程的初值问题的例子。

3.2.1 乙炔加氢

已发现在1000K下进行的乙炔非均相气相加氢生成乙烷的反应，即：

$$C_2H_2+2H_2 \longrightarrow C_2H_6$$

其反应速率可用如下速率表达式表示：

$$-r_{C_2H_2}=k[C_2H_2][H_2]^2$$

1000K下的实验研究表明反应速率常数为1×10^5mol/(L·min)。如果75%（摩尔分数）的氢和25%（摩尔分数）的乙炔混合物被加入1L的间歇反应器，而初始时刻反应器存在0.001mol的乙炔，计算乙炔的转化率达到90%需要的反应时间。

由动力学方程$-r_{C_2H_2}=k[C_2H_2][H_2]^2$，我们可以知道任意时刻乙炔的转化率，但现在问题颠倒过来了，我们想知道当乙炔的转化率达到90%时，间歇反应器反应了多少时间？这个问题可以利用求解非线性方程的对分法解决，但使用对分法

需要有任意时刻酸的转化率的数据。

所以，我们先编写一个 MATLAB 函数，其作用是能够求得任意时刻乙炔的转化率：

```
function x=g(t)
%t:时刻/min
%x:t时刻乙炔的转化率
          ca0=0.001;                   %乙炔的初始摩尔浓度/mol·L^-3
          cb0=ca0 *0.75/0.25;          %氢气的初始摩尔浓度/mol·L^-3
          [x,y]=Rk_ad(0,t,ca0,'f');    %求解初值问题
          ca=y(length(y));             %乙炔在t时刻的摩尔浓度
          x=(ca0-ca)/ca0;              %乙炔在t时刻的转化率
function y=f(t,ca)
%本函数定义微分方程的右端项
%t:时刻/min
%ca:t时刻乙炔的摩尔浓度
         ca0=0.001;                    %乙炔的初始摩尔浓度/mol·L^-3
         cb0=ca0 *0.75/0.25;           %氢气的初始摩尔浓度/mol·L^-3
         k=1e5;                        %加氢动力学方程系数
         cb=cb0-2 *(ca0-ca);           %氢气在t时刻的摩尔浓度
         y=-k *ca *cb^2;               %微分方程的右端
```

函数 $g(t)$ 里面要求求解加氢动力学微分方程，因此需要书写 $f(t,ca)$ 函数来定义加氢动力学微分方程的右端，实质就是微分方程 $dy/dx=f(x,y)$ 中的 $f(x,y)$。

有了上面的准备，就可以求得任意时刻乙酸的转化率，我们用图形来表达，MATLAB 代码如下：

```
t=1:1:30;
x=zeros(1,length(t));
for k=1:length(t)
    x(k)=g(t(k));
end
plot(t,x)
xlabel('时刻/min')
ylabel('乙炔的转化率')
```

任意时刻乙炔的转化率的图形见图 3-5：

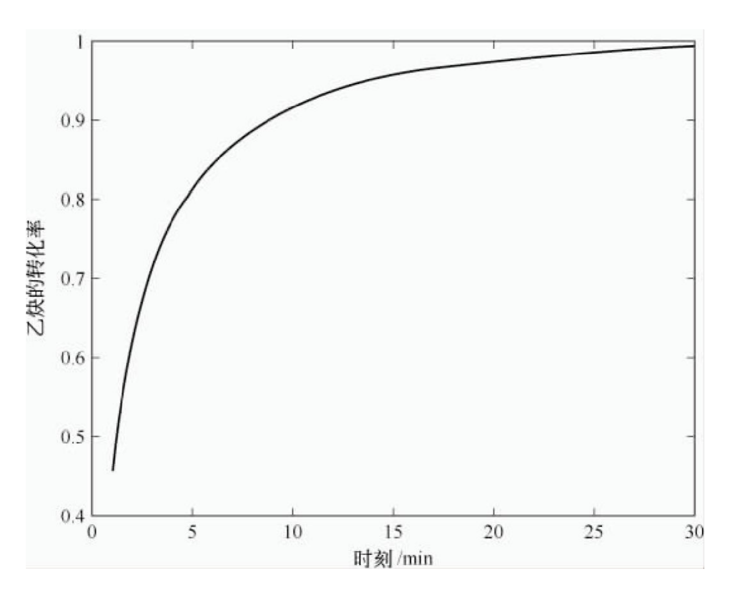


图 3-5　任意时刻乙炔的转化率

从图上 3-5 可以看出，约在 $t=10$min 时，乙炔的转化率为 90%，反应时间在 30min 以后，反应接近平衡。

自然，我们不能仅凭图形得到乙炔的转化率为 90%时需要的反应时间，我们需要精确获得这个时刻，我们用对分法求解，我们从图 3-5 看出，这一时刻应该在 [5，15] 之间。我们采用对分法求解。使用对分法前，还需要定义一函数 $p(t)$，函数 $p(t)$的零点就是乙炔的转化率达到 90%时需要的时间：

```
function y=p(t)
         y=g(t)-0.9;
```

然后利用对分法，具体的 MATLAB 程序如下：

```
t=bisec_n('p',5,15)      %乙炔的转化率为90%时需要的反应时间
```

解得：

t=8.8377

所以，在 8.8min 时，乙炔的转化率达到 90%。

3.2.2　生产乙酸乙酯

乙酸和乙醇反应生成乙酸乙酯，其反应式为：

$$C_2H_5OH+CH_3COOH=\!=\!=CH_3COOC_2H_5+H_2O$$

(A)　　(B)　　(C)　　(D)

要求的日产量为 50000kg，液相中的反应速率由下式给出：

$$-r_A=k*(c_Ac_B-c_Rc_S/K)$$

在100℃时，$k=7.93\times10^{-6}m^3/(kmol\cdot s)$　$K=2.93$。

料液中酸和醇的质量分数分别为23%和46%，酯浓度为零。酸的转化率控制在35%，物料密度基本为常数，其值是$1020kg/m^3$，反应器每天操作24h，每一生产周期中加料、出料等辅助时间总共为1h，试计算所需要的反应器体积V。

对于间歇反应器，应该有如下关系：

日产量=反应器单位时间单位体积的生产速率×反应器体积×每日操作时间

当每一生产周期结束时，应该有：

产品浓度=反应器单位时间单位体积的生产速率×(反应时间+辅助时间)

因此，问题的关键就是求得当酸的转化率达到35%时，间歇反应器需要反应多少时间？

由动力学方程$-r_A=k*(c_Ac_B-c_Rc_S/K)$，我们可以知道任意时刻酸的转化率，但现在问题颠倒过来了，我们想知道当酸的转化率达到35%时，间歇反应器反应了多少时间？解决这个问题的，可以利用求解非线性方程的对分法解决，但使用对分法需要有任意时刻酸的转化率的数据。

所以，我们先编写一个MATLAB函数，其作用是能够求得任意时刻乙酸的转化率：

```
function x=g(t)
%t:时刻/h
%x:t时刻乙酸的转化率
        rou=1020;                      %物料密度
        cb0=rou*0.23/60;               %乙酸的初始摩尔浓度/kmol·m-3
        ca0=rou*0.46/46;               %乙醇的初始摩尔浓度/kmol·m-3
        [x,y]=Rk_ad(0,t*3600,ca0,'f');    %求解初值问题
        ca=y(length(y));               %乙醇在t时刻的摩尔浓度
        cb=cb0-(ca0-ca);               %乙酸在t时刻的摩尔浓度
        x=(cb0-cb)/cb0;                %乙酸在t时刻的转化率
function y=f(t,ca)
%本函数定义微分方程的右端项
%t:时刻/h
%ca:t时刻乙醇的摩尔浓度
       rou=1020;                       %物料密度
       ca0=rou*0.46/46;                %乙醇的初始摩尔浓度/kmol·m-3
       cb0=rou*0.23/60;                %乙酸的初始摩尔浓度/kmol·m-3
       cr0=0;                          %酯的初始浓度
       cs0=rou*(1-0.23-0.46)/18;       %水的初始浓度
```

```
k=7.93e-6;                    %酯化动力学方程系数
K=2.93;                       %酯化动力学方程系数
cb=cb0-(ca0-ca);              %乙酸在 t 时刻的摩尔浓度
cr=cr0+(ca0-ca);              %酯在 t 时刻的摩尔浓度
cs=cs0+(ca0-ca);              %水在 t 时刻的摩尔浓度
y=-k*(ca*cb-cr*cs/K);         %微分方程的右端
```

函数 $g(t)$里面要求求解酯化动力学微分方程，因此需要书写 $f(t,ca)$函数来定义酯化动力学微分方程的右端，实质就是微分方程 $dy/dx=f(x,y)$中的 $f(x,y)$。

有了上面的准备，就可以求得任意时刻乙酸的转化率，我们用图形来表达：

```
t=0.1:0.1:10;
x=zeros(1,length(t));
for k=1:length(t)
    x(k)=g(t(k));
end
plot(t,x)
xlabel('时刻/h')
ylabel('乙酸的转化率')
```

任意时刻乙酸的转化率图形见图 3-6：

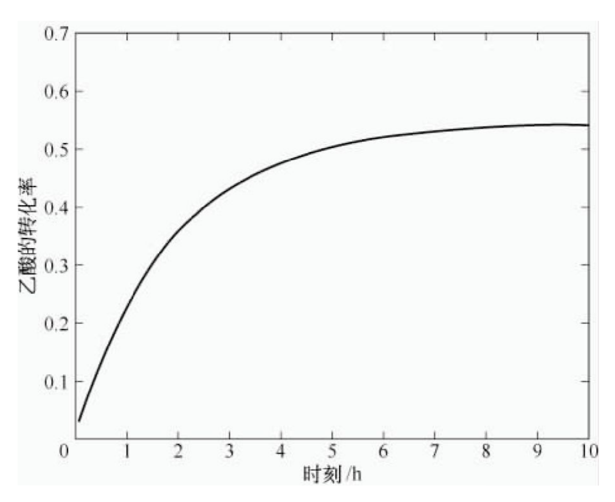


图 3-6　任意时刻乙酸的转化率

从图上 3-6 可以看出，约在 $t=2$h 时，乙酸的转化率为 35%，反应时间在 9 小时以后，反应接近平衡。

自然，我们不能仅凭图形得到乙酸的转化率为 35%时需要的反应时间，需要精

确获得这个时刻，我们用对分法求解，我们从图3-6看出，这一时刻应该在［1，3］之间。在获得乙酸的转化率为35％时需要的反应时间后，一切问题迎刃而解，就可以顺利求得需要的反应器体积。请注意如果要获得乙酸的转化率为35％时需要的反应时间，还需要定义一个函数 $p(t)$，用于求函数 $p(t)=g(t)-0.35$ 的零点。自然需要把上面所有函数都单独保存在MATLAB安装目录下的work子目录里。

具体的MATLAB程序如下：

```
rou=1020;                              %物料密度
cb0=rou *0.23/60;                      %乙酸的初始摩尔浓度/kmol·m^-3
t=bisec_n('p',1,3);                    %乙酸的转化率为35%时需要的反应时间
cr=cb0 *0.35;                          %乙酸转化率为35%时酯的浓度
v=(50000/(24 *88))/(cr/(t+1))          %反应器体积
function y=p(t)
    y=g(t)-0.35;
```

计算结果是：

v=51.5276

因此，需要的反应器体积是51.5m^3。

上述问题从化学角度说很简单，但具体计算上却很麻烦，如果不会编制MATLAB程序求解，只有手动自行积分，但该动力学方程比较复杂，虽然也能积分出来，但容易出错。

3.3 一阶常微分方程组的初值问题

考虑一阶常微分方程组的初值问题：

$$\frac{dy_i}{dx}=f_i(x,y_1,y_2\cdots,y_1) \qquad y_i(x_0)=y_{i0},i=1,2,\cdots,l$$

若把其中的未知函数，方程的右端项都表达成向量形式：

$$Y=(y_1,y_2,\cdots,y_l)',F=(f_1,f_2,\cdots,f_l)'$$

初值条件表为：

$$Y(x_0)=Y_0=(y_{10},y_{20},\cdots,y_{l0})'$$

那么方程组可以写成：

$$\frac{dY}{dx}=F(x,Y)$$

$$Y(x_0)=Y_0,i=1,2,\cdots,l$$

上述向量化的常微分方程组初值问题在形式上和常微分方程的初值问题在形式上相似，求解常微分方程初值问题的嵌入式龙格-库塔方法完全适用于常微分方程组初值问题的求解，唯一需要注意的是，对于实现嵌入式龙格-库塔方法的函数：

```
Rk_ad(a,b,y0,f_name,tolerance)
```

用于求解常微分方程组的初值问题时，y_0 是由（y_{10}，y_{20}，…，y_{l0}）组成的向量，函数 f_name［用于定义 $f(x,y)$］定义的是一个函数向量，即 $F(x,Y)=(f_1(x,Y),f_2(x,Y),\cdots,f_l(x,Y))'$。

下面我们举两个例子，该例子包含常微分方程组的初值问题。

3.3.1　平行反应各物质浓度与时间的关系曲线

假定在间歇反应器中有下列动力学体系：

$A+B \longrightarrow C$　　　　$R_1=k_1{c_A}^{1/2}c_B$

$B+C \longrightarrow D$　　　　$R_2=k_2{c_C}^{1/2}c_B$

做出各物质的浓度 c_A、c_B、c_C、c_D对时间的函数图形。给定 $a_0=10\text{mol/m}^3$，$b_0=2\text{mol/m}^3$，$c_0=d_0=0$；$k_1=k_2=0.02\text{m}^{3/2}/(\text{mol}^{1/2}\cdot\text{s})$，求多长时间后，限制反应物被消耗掉90％。

根据间歇反应器内的物料衡算关系：

$dc_A/dt=-k_1{c_A}^{1/2}c_B$

$dc_B/dt=-k_1{c_A}^{1/2}c_B-k_2c^{1/2}c_B$

$dc_C/dt=k_1{c_A}^{1/2}c_B-k_2c^{1/2}c_B$

$dc_D/dt=k_2{c_C}^{1/2}c_B$

解决上述两个问题的关键是求解上述常微分方程组。我们先书写 MATLAB 函数，定义上述微分方程组的右端项 $F(x,Y)$：

```
function yf=f(t,y)
k1=0.02;
k2=k1;
yf(1)=-k1 *sqrt(y(1))*y(2);
yf(2)=-k1 *sqrt(y(1))*y(2)- k2 *sqrt(y(3))*y(2);
yf(3)=k1 *sqrt(y(1))*y(2)- k2 *sqrt(y(3))*y(2);
yf(4)=k2 *sqrt(y(3))*y(2);
```

定义上述微分方程组的右端项 $F(x,Y)$的工作完成后，我们可以画出 c_A、c_B、c_C、c_D对时间的函数图形，具体的 MATLAB 程序如下：

```
cA0=10;cB0=2;cC0=0;cD0=0;
y0=[cA0 cB0 cC0 cD0];
[t,y]=Rk_ad(0,50,y0,'f');
plot(t,y(:,1) ,'-*b',t,y(:,2),'-or' ,t,y(:,3),'-hk',t,y(:,4),'-dg')
xlabel('时间/s')
ylabel('浓度/mol/m3')
legend('cA','cB','cC','cD','Location','best')
```

做出的图形见图3-7：

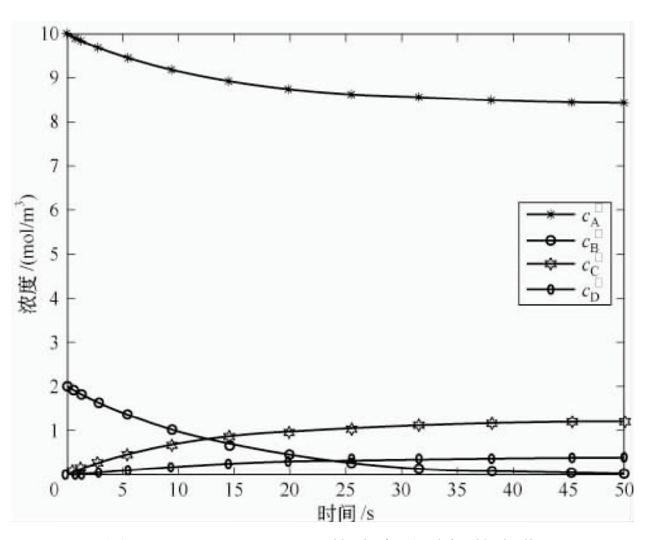


图3-7　A、B、C、D的浓度随时间的变化

很明显，限制反应物是B，为了解决求多长时间后，限制反应物被消耗掉90%，我们必须实现定义一个函数，它能够求得（任意时刻B的浓度－0.1）的值，该函数如下：

```
function B=g(t)
cA0=10;cB0=2;cC0=0;cD0=0;
y0=[cA0 cB0 cC0 cD0];
[t,y]=Rk_ad(0,t,y0,'f');
B=y(size(y,1),2)-0.1;
```

从图3-7看出，在50s时，B基本消耗殆尽，所以我们把求解区间定为[0 50]，求多长时间后，限制反应物B被消耗掉90%的MATLAB程序如下：

t=bisec_n('g',0,50)

结果是：

t=38.5389

38.5秒后，B被消耗掉90%。

3.3.2　串联反应各物质浓度与时间的关系曲线

MATLAB自带的求解常微分方程（组）的初值问题的函数是ode45函数，其基本使用格式如下：

```
[T,Y]=ode45(ODEFUN,TSPAN,Y0)
```

ode45函数采用的方法类似于嵌入式龙格-库塔方法，因此求解精度与嵌入式龙

格-库塔方法相同。ODEFUN是一字符串，代表函数文件名，该函数文件定义常微分方程（组）的右端项 $F(x,Y)$，需要注意的是，该函数必须返回函数列向量 $(f_1(x,Y), f_2(x,Y), \cdots, f_l(x,Y))'$。TSPAN代表求解区间，它是一个形如 $[x_0 \quad x_n]$ 二元行向量，x_0 代表求解的初始点，x_n 代表求解终点，Y0是一向量，代表初始条件 $(y_{10}, y_{20}, \cdots, y_{l0})$。ode45函数返回T和Y两个值，其意义与Rk _ad 函数相同。T是一列向量，第一个分量是 x_0，最后一个分量是 x_n，其余是介于 (x_0, x_n) 的数值，Y是列向量或矩阵，当求解常微分方程的初值问题时，返回列向量，当求解常微分方程组的初值问题时，返回矩阵。矩阵的每列代表与列向量T相对应的函数值。

考虑下面的反应序列：

$$A \xrightarrow{k_A} B \xrightarrow{k_B} C \xrightarrow{k_C} D \xrightarrow{k_D} E$$

$k_A = 1h^{-1}$，$k_B = 0.5h^{-1}$，$k_C = 0.25h^{-1}$，$k_D = 0.125h^{-1}$，A的初始浓度为 $10mol/m^3$，其余各组分初始浓度为0，做出A、B、C、D和E对时间的函数图形。

根据反应动力学，有：

$dc_A/dt = -k_A c_A$

$dc_B/dt = k_A c_A - k_B c_B$

$dc_C/dt = k_B c_B - k_C c_C$

$dc_D/dt = k_C c_C - k_D D_D$

$dc_E/dt = k_D D_D$

我们先定义上述常微分方程组的右端项：

```
function y=f(t,c)
kA=1;kB=0.5;kC=0.25 ;kD=0.125;
y(1)=-kA *c(1); y(2)=kA *c(1)- kB *c(2);
y(3)=kB *c(2)- kC *c(3);y(4)=kC *c(3)- kD *c(4);
y(5)=kD *c(4);
y=y';
```

定义上述微分方程组的右端项 $F(x,Y)$ 的工作完成后，我们可以画出做出A、B、C、D和E对时间的函数图形，具体的MATLAB程序如下：

```
a0=10;b0=0;c0=0;d0=0;e0=0;
y0=[a0 b0 c0 d0 e0];
[t,y]=ode45('f',[0 10],y0);
plot(t,y(:,1) ,'- *b',t,y(:,2),'-or' ,t,y(:,3),'-hk',t,y(:,4),'-dg',t,y(:,5),'-<m')
xlabel('时间/h')
ylabel('浓度/mol/m3')
legend('a','b','c','d','e','Location','best')
```

做出的图形见图 3-8：

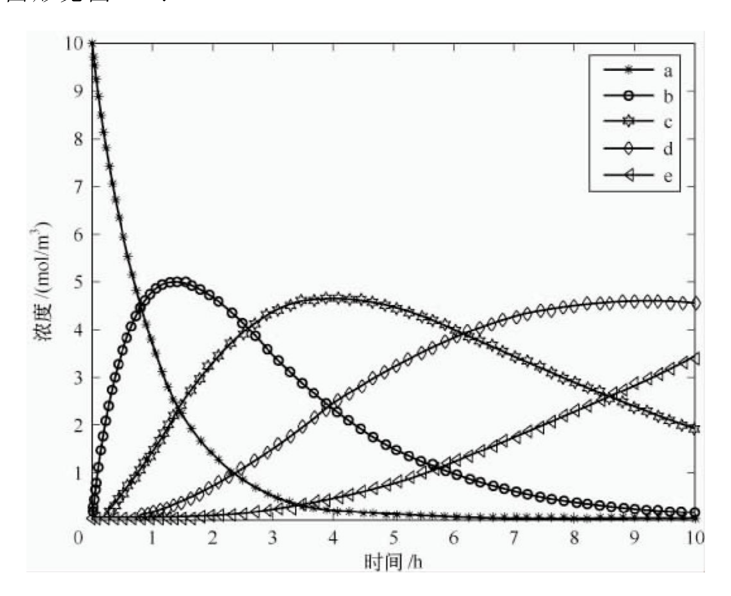


图 3-8　A、B、C、D 和 E 的浓度随时间的变化



4.1 线性方程组的求解

大量的科学与工程计算问题可以归结为数值求解线性方程组，对化学化工中的计算，自然也不例外。我们在初中就学过如何求解二元和三元线性方程组，那时我们学习了用代入法或消元法求解线性方程组。

4.1.1 Gauss 主元消去法

原则上说，使用代入法或消元法可以求解任意的线性方程组，但一旦线性方程组的未知数变多，使用初中讲授的代入法或消元法，极易出错，因此我们这里需要采用新的求解方法。我们这里将要讲授的方法，是消元法的进一步的发展，被称为 Gauss 主元消去法。

考虑一般的线性方程组：

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{12} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \qquad ①$$

简写为：$AX=B$。A 是系数矩阵，B 是右端向量。

如果能够将线性方程组①变换为上三角形式的同解线性方程组：

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix} \qquad ②$$

简写为：$UX=E$。

则求线性方程组①的解的过程将至为简单！

U 是一上三角阵，即矩阵 U 的对角线（不包含对角线）以下的各元素均为 0。很明显。线性方程组②，即 $UX=E$ 非常容易求解，只需要从方程组②的最后一行开始求解，先求出 x_n，然后依次求出 x_{n-1}、x_{n-2}…直至最后求出 x_1。

将线性方程组①变换为同解的线性方程组②，然后求解与①同解的线性方程组②，从而得到线性方程组①的解的过程，称为 Gauss 消去法。Gauss 消去法求解线

性方程组①包括两个过程，一个是消去过程，一个是回代过程。

我们将线性方程组①的系数矩阵 A 和右端向量 B 合并，写成增广矩阵 AB：

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{bmatrix}$$

消去过程：

（1）增广矩阵 AB 第 2 行－增广矩阵 AB 第 1 行 $\times a_{21}/a_{11}$，得到的行向量作为第 2 行；增广矩阵 AB 第 3 行－增广矩阵 AB 第 1 行 $\times a_{31}/a_{11}$，得到的行向量作为第 3 行；……增广矩阵 AB 第 n 行－增广矩阵 AB 第 1 行 $\times a_{n1}/a_{11}$，得到的行向量作为第 n 行；经过上述步骤，增广矩阵 AB 化为增广矩阵 AB'：

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a'_{22} & \cdots & a'_{2n} & b'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & a'_{n2} & \cdots & a'_{nn} & b'_n \end{bmatrix}$$

至此完成第 1 次消去操作。

（2）仿照步骤 1，在增广矩阵 AB'中用圆角矩形扩起来的子矩阵中进行第 2 次消去操作：

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a'_{22} & \cdots & a'_{2n} & b'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & a'_{n2} & \cdots & a'_{nn} & b'_n \end{bmatrix}$$

得到新的增广矩阵 AB"：

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a'_{22} & \cdots & a'_{2n} & b'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & a''_{nn} & b''_n \end{bmatrix}$$

（3）一共进行 n 次消去操作，最终得到增广矩阵 $AB^{(n)}$，其中的子矩阵 A（系数矩阵），化为上三角阵。

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a'_{22} & \cdots & a'_{2n} & b'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(n)} & b_n^{(n)} \end{bmatrix}$$

回代过程：从增广矩阵 $AB^{(n)}$ 的最后一行开始求解，先求出 x_n，然后依次求出 x_{n-1}、x_{n-2}…直至最后求出 x_1。

很明显，要完成 Gauss 消去法，必须要求系数矩阵 A 的对角线元素都不为 0，此外为了降低误差，保证求解过程的精度，一般希望系数矩阵 A 的对角线元素的绝对值越大越好，基于这两点，人们提出了更有效、计算精度更高的 Gauss 主元消去法。

Gauss 主元消去法分为列主元消去法和全主元消去法。

Gauss 列主元消去法是指在执行第 k 次消去操作之前，在当前增广矩阵的第 k 列的第 k 行到第 n 行寻找绝对值最大的元素，然后将包含最大绝对值元素的那一行与第 k 行进行互换，然后执行第 k 次消去操作。这个绝对值最大元素被称为主元，由于这个主元是在某一列当中选取的，该主元又称为列主元，我们把这种方法称为列主元消去法。选列主元的过程，实际上是把线性方程组中的两个线性方程的位置进行了交换，我们可以很明显地看出，这并不会使得线性方程组的解发生改变。

Gauss 全主元消去法是指在执行第 k 次消去操作之前，在当前增广矩阵的第 k 列到第 n 列和第 k 行到第 n 行构成的子矩阵中寻找绝对值最大的元素，设在此子矩阵找到的绝对值最大元素在第 m 行、第 n 列，然后将当前增广矩阵的第 m 行与第 k 行互换，再将第 n 列与第 k 列互换，然后执行第 k 次消去操作。由于寻找绝对值最大的元素是在相应子矩阵内的全部元素中寻找的，这时找到的主元称为全主元，该方法称为全主元消去法。我们在前面提到，将增广矩阵内的各行互换，相当于线性方程组中的两个线性方程的位置进行了交换，不会使得线性方程组的解发生改变。但互换增广矩阵的两列，如交换第 j 列和第 k 列，相当于将线性方程组的解向量 X 的第 j 个分量和第 k 个分量互换，这可能导致最后求得的解向量 X 与原始线性方程组的解向量在顺序上不一致，在具体编写 Gauss 全主元消去法的 MATLAB 程序时，应当书写额外的代码，以保证最后得到的解向量与原始线性方程组的解向量在顺序上一致。

Gauss 全主元消去法比列主元消去法具有更高精度。

Gauss 全主元消去法的 MATLAB 函数代码如下：

```
function x=gauss_e(a,b)
% 利用高斯全主元消去法求解线性方程组 ax=b
% a:系数矩阵,b:右端列向量,x:解向量
a=[a b];
n=size(a,1);                          %获取增广矩阵的行数
x=zeros(n,1);
L=1:n;                                %解向量的正常顺序,[1 2 3…n]
for k=1:n-1
    [c1,x1]=max(abs(a(k:n,k:n)));
    [c2,y1]=max(c1);
    x1=x1(y1);                        %以上语句在子矩阵中选全主元
    x1=x1+(k-1);
```

```
        y1=y1+(k-1);                                    %以上语句确定主元
                                                         在整个增广矩阵的位置
        c=a(x1,:);a(x1,:)=a(k,:);a(k,:)=c;              %换行操作
        c=a(:,y1);a(:,y1)=a(:,k);a(:,k)=c;              %列交换操作
        c=L(y1);L(y1)=L(k);L(k)=c;                      %记忆列交换操作
        % 消元
        for m=k+1:n
            a(m,k+1:n+1)=a(m,k+1:n+1)-a(k,k+1:n+1)/a(k,k)*a(m,k);
        end
    end
    % 回代
    for k=n:-1:1
        x(k)=(a(k,n+1)-dot(a(k,k+1:n),x(k+1:n)'))/a(k,k);
    end
    [k,m]=sort(L);                                      %找到解向量中各分量的位置
    x=x(m);                                             %获得正常顺序的解
```

对于执行 Gauss 全主元消去法的 gauss_e（a，b）函数，最需要注意的就是如何保证最后得到的解向量与原始线性方程组的解向量在顺序上一致。为此，函数内部有一条赋值语句：

L=1:n，即 L 被赋值为行向量［1 2…n］

因为原始线性方程组的解向量的排列顺序是［x_1 x_2…x_n］，所以 L 向量［1 2…n］是解向量［x_1 x_2…x_n］的一个映象或对应物。假设我们在选全主元的过程中，将第 2 列与第 3 列进行了交换，将来的解出来解向量就会变成：［x_1 x_3 x_2…x_n］。

为了标记这一变动，将行向量的第 2 个分量与第 3 个分量进行交换，得到：［1 3 2…n］。

就采用上面的方式，在选全主元过程中不断根据列交换的情况改变向量 L，直到选全主元过程结束。

假设求解的线性方程组是一个四元方程组，选全主元过程结束后，得到的向量 L 为：［3 1 2 4］。

回代过程求的解向量形式就会是［x_3 x_1 x_2 x_4］，我们设这一向量为 X'。

如何恢复解向量的正常顺序呢？这就需要用到 MATLAB 的 sort 函数，sort 函数的作用是将一个向量的各分量按照从小到大排序，并返回经过排序的向量的各分量在原来未排序向量里的位置。具体使用方式如下：

```
[k,m]=sort([3 1 2 4]);
k
m
```

结果是：

k=1　2　3　4

m=2　3　1　4

k 是经过排序的向量，m 是反映 k 向量各分量在原来的［3 1 2 4］向量中位置的向量。我们这里需要的不是经过排序的向量 k，真正对我们有用的是位置向量 m。

我们经过回代，得到解向量［x_3 x_1 x_2 x_4］，我们利用位置向量 m，可以把解向量［x_3 x_1 x_2 x_4］重新排列为［x_1 x_2 x_3 x_4］，具体如下：

$X=X'(m)$

上述赋值语句的意思就是依次以向量 m 的各分量数值作为向量 X 的索引值或角标，依次取出 x'_2、x'_3、x'_1 和 x'_4，组成一向量，赋给 X。请注意，x'_2、x'_3、x'_1 和 x'_4 就是 x_1、x_2、x_3 和 x_4。由此，我们完成了保证最后得到的解向量与原始线性方程组的解向量在顺序上一致的操作。

上述 gauss_e（a，b）函数中还使用了 max 函数，max 函数是 MATLAB 的固有函数，当 max 函数作用于向量时，它返回向量里所有分量中最大的分量，并且返回该最大分量的所在的位置。如：

```
≫[m n]=max([3 4 2 1])
m=4
n=2
```

当 max 函数作用于矩阵时，返回矩阵每列中的最大值，组成一个行向量，并且返回每列最大值所在的行数，如：

```
≫a=[8 1 6;3 5 7;4 9 2];
≫[m n]=max(a)
m=8   9   7
n=1   3   2
```

MATLAB 本身采用反除算符（\）求解线性方程组 $AX=B$，具体求解格式如下：

$X=A\backslash B$

采用的算法仍然是 Gauss 主元消去法。

4.1.1.1　金属涂覆操作

图 4-1 所示的操作单元是在金属部件表面涂覆上一种聚合物。用含有这种聚合物的溶液来处理金属，之后将溶剂蒸发掉，这种聚合物即被涂覆在金属表面。

金属部件以 76kg/min 的速率进入涂覆器，请求出物流①、③和④的流率（kg/min），图中分率为质量分率。

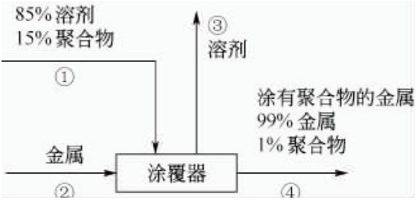


图 4-1　涂覆操作单元

本问题是稳态流动条件下的物料衡算问题，系统处于稳态流动条件时的物料衡算式是：

物料的输入速率=物料的输出速率

进出该操作单元的物料包括溶剂、聚合物和金属，分别对这三种物料列出物料衡算式子，可得3个独立方程，而问题问物流①、③和④的流率，3个独立方程，3个未知数，故问题可解决。

设物流①、③和④的流率分别为 $F1$、$F3$ 和 $F4$，得到3元一次方程组：

$F1\times0.85-F3=0$

$F1\times0.15-F4\times0.01=0$

$F4\times0.99=76$

求解该线性方程组的MATLAB语句如下：

```
a=[0.85 -1 0;0.15 0 -0.01;0 0 0.99];
b=[0;0;76];
F=a\b
```

结果是：

```
     5.1178
F=   4.3502
    76.7677
```

所以物流①、③和④的流率分别是5.1178kg/min、4.3502kg/min和76.7677kg/min。

注意在上述语句中b是列向量。

用函数gauss_e函数的结果是：

```
F=gauss_e(a,b)
     5.1178
F=   4.3502
    76.7677
```

4.1.1.2 生产高浓度硝酸

用氨气生产硝酸的工艺中，生产出来的是60%的硝酸水溶液。因为在共沸点时硝酸的浓度是68%，所以不能用普通精馏方法获得浓度为99%的硝酸。但是，如果向稀硝酸溶液添加硝酸镁的话，就可以通过精馏的方法获得浓度大于68%的硝酸，然后再对硝酸溶液进行精馏操作，最后获得高浓度硝酸，有关的工艺流程示意图见图4-2。

试求物流③至物流⑨的流率，图中分率为质量分率。

这仍然是物料衡算问题，需要根据问题来划定衡算范围和衡算基准。本问题要求求7个未知质量流率，我们设为 $F3$、$F4$ 直到 $F9$。下面来看看如何建立7个独立的物料衡算方程，以解决提出的问题。这里，我们统一以单位时间作为衡算基准。

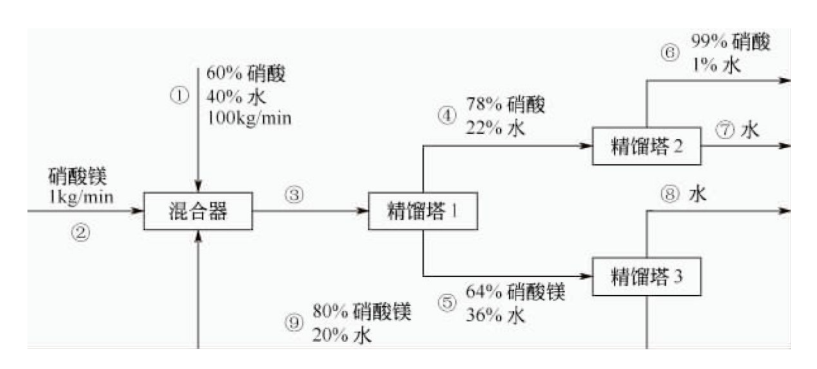


图 4-2　生产高浓度硝酸的工艺流程

以精馏塔 2 作为衡算范围，对硝酸和水作物料衡算，得到：

$F4 \times 0.78 - F6 \times 0.99 = 0$

$F4 \times 0.22 - F6 \times 0.01 - F7 = 0$

以精馏塔 3 作为衡算范围，对硝酸镁和水作物料衡算，得到：

$F5 \times 0.64 - F9 \times 0.8 = 0$

$F5 \times 0.36 - F8 - F9 \times 0.2 = 0$

以混合器和精馏塔 1 作为衡算范围，对硝酸和水作物料衡算，得到：

$F4 \times 0.78 = 100 \times 0.6$

$F4 \times 0.22 + F5 \times 0.36 - F9 \times 0.2 = 100 \times 0.4$

以精馏塔 1 作为衡算范围，做总物料衡算，得到：

$F3 - F4 - F5 = 0$

以上获得了 7 个独立方程，含 7 个未知数，可以求解，语句如下：

```
a2=[0 0.78 0 -0.99 0 0 0;0 0.22 0 -0.01 -1 0 0];
a3=[0 0 0.64 0 0 0 -0.8;0 0 0.36 0 0 -1 -0.2];
a_h_1=[0 0.78 0 0 0 0 0;0 0.22 0.36 0 0 0 -0.2];
a1=[1 -1 -1 0 0 0 0];
a=[a2;a3; a_h_1;a1];
b=[0 0 0 0 100 *0.6 100 *0.4 0]';
F=a\b
['F6+F7+F8=' num2str(sum(F(4:6)))]
```

结果是：

F=192.3077　76.9231　115.3846　60.6061　16.3170　23.0769　92.3077

ans=F6+F7+F8=100

由于整个系数矩阵 a 是个 7×7 的矩阵，比较大，一口气写下来很不方便。所以就按照 4 个衡算范围依次写出 $a2$、$a3$、a_h_1 和 $a1$ 子矩阵，然后将这些矩阵

合并为系数矩阵 a。请注意对右端向量 b 赋值时采用了矩阵转置算符（'），将行向量转置为列向量。

最后一条语句['F6＋F7＋F8＝' num2str(sum(F(4:6)))]是一个校验步骤。因为本问题涉及的未知数较多，为了防止出错，所以设一校验步骤。从图4-2可以很明显地看出，物流⑥、⑦和⑧的质量流率之和应该等于物流①的质量流率，即100kg/min，而这个关系并未直接用于求解各物流流率的过程中，因此可以用这个关系来校核计算的正确性。'F6＋F7＋F8＝'是一字符串常数，在MATLAB中，用单引号（'）把字母或数字括起来，字母和数字就成了字符串常数。MATLAB将字符串常数视为一行向量，每个分量是该字符串常数的一个字符。sum(F(4:6))自然是计算物流⑥、⑦和⑧的质量流率之和，结果是一数值。num2str是MATLAB的固有函数，它的作用是把数值转换为字符串常数。因此num2str(sum(F(4:6)))实际就是字符串常数'100'，既然字符串常数被看作行向量，自然字符串常数能够进行合并，所以['F6＋F7＋F8＝' num2str(sum(F(4:6)))]就是进行两个字符串合并的，字符串合并时，要用中括号把它们括起来。

4.1.2 LU分解

在上面生产高浓度硝酸，求物流③至物流⑨的流率的例子中，我们考察系数矩阵 a：

$$a=\begin{matrix} 0 & 0.7800 & 0 & -0.9900 & 0 & 0 & 0 \\ 0 & 0.2200 & 0 & -0.0100 & -1.0000 & 0 & 0 \\ 0 & 0 & 0.6400 & 0 & 0 & 0 & -0.8000 \\ 0 & 0 & 0.3600 & 0 & 0 & -1.0000 & -0.2000 \\ 0 & 0.7800 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2200 & 0.3600 & 0 & 0 & 0 & -0.2000 \\ 1.0000 & -1.0000 & -1.0000 & 0 & 0 & 0 & 0 \end{matrix}$$

我们可以看出，矩阵 a 含有较多的零元素，零元素的个数占整个矩阵 a 元素个数的65％。这种含零元素比较多的矩阵称为稀疏矩阵，系数矩阵是稀疏矩阵的线性方程组被称为稀疏方程组。这类方程组经常出现在实际问题的规划、优化和大型化工厂的物料与能量衡算时。求解稀疏方程组，使用高斯主元消去法当然是可以的，比如上面的例子就使用了高斯主元消去法求解稀疏方程组。但如果对于大型稀疏方程组，再使用高斯消去法就不是很合适了，因为高斯消去法运算量比较大，占用机时较长，也需要较大的内存空间。求解大型稀疏线性方程组的较好方法是LU分解法和直接迭代法。我们这里介绍LU分解法，对于直接迭代法，请读者参看有关数值分析教材。

LU分解法计算量小，占用计算机内存小，便于储存，是目前应用最广泛的一种线性方程组的求解方法。所谓LU分解，顾名思义，就是把线性方程组 $AX=B$ 的系数矩阵 A 分解为下三角阵 L 和上三角阵 U 的乘积：$LUX=B$。

下三角阵 L 的形式为：

$$
\begin{matrix}
\alpha_{11} & & & \\
\alpha_{21} & \alpha_{22} & & \\
\vdots & \vdots & \ddots & \\
\alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{nn}
\end{matrix}
$$

上三角阵 U 的形式为：

$$
\begin{matrix}
\beta_{11} & \beta_{12} & \cdots & \beta_{1n} \\
 & \beta_{22} & \cdots & \beta_{2n} \\
 & & \ddots & \vdots \\
 & & & \beta_{nn}
\end{matrix}
$$

令 $Y=UX$，则得到关于 Y 的线性方程组 $LY=B$，利用回代步骤，求得 Y 向量。

因为 $UX=Y$，Y 已经求得，所以再次利用回代步骤，求得线性方程组的最终解 X。

因此，问题的关键就是如何将系数矩阵 A 分解为 LU 之积，由于对系数矩阵 A 分解为 LU 之积的算法比较复杂，我们这里直接介绍如何利用 MATLAB 对系数矩阵 A 进行 LU 分解。MATLAB 使用 lu 函数对系数矩阵 A 进行 LU 分解，使用格式如下：

$$[L\ U\ P]=\mathrm{lu}(A)$$

L 是对角线元素全为 1 的下三角阵，U 是上三角阵，P 是交换矩阵，其元素是 0 或 1。这三个矩阵和系数矩阵 A 的关系是：$PA=LU$。

为了使用上述形式的 LU 分解求解线性方程组 $AX=B$，我们在上述分解形式两边都乘以未知向量 X，得到 $PAX=LUX$。

由矩阵乘法的分配律 $P(AX)=LUX$ 得到 $PB=LUX$。因此，求线性方程组 $AX=B$ 的解的问题，就转化成求线性方程组 $LUX=PB$ 的解的过程，上述方程组是 LU 分解的形式，可以利用两次回代过程求解。

具体的 MATLAB 函数如下：

```
function x=lu_p(a,b)
% 使用 lu 分解求解线性方程组 ax=b
% a:系数矩阵,b:右端向量
if size(b,1)==1                          %如果向量 b 是行向量,则进行转置
   b=b';
end
[l,u,p]=lu(a);b=p*b;                     %进行 LU 分解和问题转换
y=zeros(1,size(a,1));x=y;
for k=1:size(a,1)
     y(k)=b(k)-dot(l(k,1:k-1),y(1:k-1));
end
for k=size(a,1):-1:1
     x(k)=(y(k)-dot(u(k,k+1:size(a,1)),x(k+1:size(a,1))))/u(k,k);
end
x=x';
```

在上面生产高浓度硝酸，求物流③至物流⑨的流率的例子中，利用LU分解求解的语句如下：

```
a2=[0 0.78 0 -0.99 0 0 0;0 0.22 0 -0.01 -1 0 0];
a3=[0 0 0.64 0 0 0 -0.8;0 0 0.36 0 0 -1 -0.2];
a_h_1=[0 0.78 0 0 0 0 0;0 0.22 0.36 0 0 0 -0.2];
a1=[1 -1 -1 0 0 0 0];
a=[a2;a3; a_h_1;a1];
b=[0 0 0 0 100 *0.6 100 *0.4 0]';
F=lu_p(a,b)
['F6+F7+F8=' num2str(sum(F(4:6)))]
```

结果是：

F=192.3077　76.9231　115.3846　60.6061　16.3170　23.0769　92.3077

ans=F6+F7+F8=100

4.1.3 病态现象

有一类线性方程组被称为病态线性方程组，病态线性方程组的含义在于：对于线性方程组 $AX=B$，如果系数矩阵 A 或右端向量 B 有非常微小的改变，将会引起解向量 X 的巨大变化，这样的现象称为病态现象，具有这样的性质的线性方程组，被称为病态线性方程组。病态现象是系数矩阵本身的特性，与所用的计算工具和计算方法无关。但是，实际计算过程中的病态现象却是通过所用的计算工具表现出来的。例如，计算机的字长（字长是指计算机内部参与运算的数的位数。字长直接反映了一台计算机的计算精度）越长，病态现象在程度上就会相对的减轻。

线性代数上用矩阵的条件数衡量系数矩阵是否病态，系数矩阵条件数越大，该线性方程组病态程度越高。MATLAB采用cond函数计算一个矩阵的条件数，具体格式如下：

c=cond(A)　%求矩阵 A 的2-范数的条件数，即 A 的最大奇异值和最小奇异值的商。

由于我们在使用Gauss主元消去法的过程中，要进行加减乘除的各种运算，运算的中间的结果往往会出现小数位数无穷多的数字，因为计算机存储数字的位数是有限的，因此必须进行数字的舍入，将会出现舍入误差，如果线性方程组是病态的，尽管方法和计算没有任何错误，只是在消去过程中对小数后面若干位进行了舍入，最后得到解向量 X 将会和真实解向量 X 有巨大差别。MATLAB采用15位双精度数据进行计算，降低了舍入误差的影响，有效地防止了病态问题。

4.1.4　矛盾线性方程组

所谓矛盾方程组是指方程数目大于未知数个数的方程组，从严格的数学意义上说，这种方程组是没有解的，但从另外一个角度说，这种方程组存在最小二乘解。矛盾线性方程组的最小二乘解在实际中具有广泛的应用，特别是在直线拟合方面具有广泛应用。

我们通过下面的例子说明什么是矛盾线性方程组的最小二乘解。

比如有下列二元一次方程组：

$2x_1+3x_2=5$

$x_1+2x_2=8$

$6x_1+7x_2=10$

使得函数：

$$F(x_1,x_2)=(2x_1+3x_2-5)^2+(x_1+2x_2-8)^2+(6x_1+7x_2-10)^2$$

取得最小值的 x_1、x_2 就是上述矛盾二元一次方程组的最小二乘解。

很明显，绝对不会存在 x_1、x_2 同时满足上面三个方程，或者说绝对不会存在 x_1、x_2 使得 $F(x_1,x_2)=0$，由于 $F(x_1,x_2)\geqslant 0$，所以我们转而求其次，找合适的 x_1、x_2 使得平方和形式的函数 $F(x_1,x_2)$ 达到最小。这就是矛盾线性方程组最小二乘解的意义。

那么如何去求矛盾线性方程组的最小二乘解呢？要不要我们去求函数 $F(x_1,x_2)$ 的最小值点呢？幸运的是，MATLAB 提供了直接求最小二乘解的方法。

对于矛盾线性方程组 $AX=B$，它的最小二乘解就是 $X=A\setminus B$。同样是使用反除（\）算符，和求普通线性方程组的方法一样。

比如求上述矛盾二元线性方程组的最小二乘解的 MATLAB 语句是：

```
a=[2 3;1 2;6 7];
b=[5 8 10]';
x=a\b
F=(norm(a *x-b))^2
```

结果是：

```
x=  -5.0952
     5.7381
```

最后一个语句是求平方和函数 F 在最小值点的函数值，在最小二乘的意义上，也称为残差。norm 函数是 MATLAB 的固有函数，用于求向量的 2-范数。

我们看两个个实际例子，体会一下 MATLAB 解矛盾线性方程组在实际中的应用。

4.1.4.1　确定反应速率常数和反应级数

已知某反应的速率方程可表示为：

$$r=k[A]^{\alpha}[B]^{\beta}[C]^{\gamma}$$

请根据下列实验数据，见表4-1。确定速率常数 k 和反应级数 α、β 和 γ。

表 **4-1**　某反应的动力学数据

$r/(10^{-5}\,\mathrm{mol\cdot dm^{-3}\cdot s^{-1}})$	5.0000	1.6667	5.0000	5.0000	2.5000	9.8432	14.1000	2.8200
$[A]_0/(\mathrm{dm^{-3}\cdot s^{-1}})$	0.010	0.010	0.010	0.010	0.010	0.025	0.020	0.020
$[B]_0/(\mathrm{dm^{-3}\cdot s^{-1}})$	0.005	0.015	0.005	0.005	0.010	0.010	0.005	0.025
$[C]_0/(\mathrm{dm^{-3}\cdot s^{-1}})$	0.010	0.010	0.015	0.025	0.010	0.010	0.010	0.010

将速率方程两边取常用对数，进行线性化：

$$\lg r=\lg k+\alpha\lg[A]+\beta\lg[B]+\gamma\lg[C]$$

上述表格一共有8组数据，等于我们获得一个由8个线性方程构成的方程组，该线性方程组的未知向量是 $[\lg k\ \alpha\ \beta\ \gamma]$，4个未知数，8个方程，因此该方程组是矛盾线性方程组，可以用MATLAB的反除算符（\）求解。请注意，该矛盾线性方程组的系数矩阵的第1列都是数字1，第2列是8个 $\lg\ [A]_0$，第3列是8个 $\lg\ [B]_0$，第4列是8个 $\lg\ [C]_0$。

因为本问题输入的数据比较多，因此采用文件输入方式，用记事本编写如下文本文件，见图4-3：

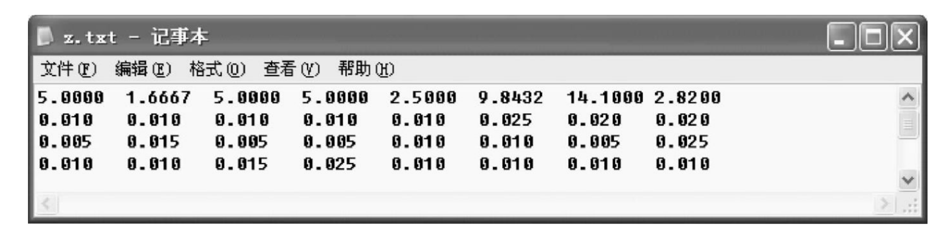


图4-3　动力学数据文件

存盘为e:\z.txt。

MATLAB语句如下：

```
load e:\z.txt
r=z(1,:);
a=z(2:4,:)';
a=[ones(8,1) log10(a)];          %生成系数矩阵
b=log10(r');                     %生成右端向量
x=a\b;                           %求解矛盾线性方程组
k=10^x(1)                        %求速率常数k
['α β γ']
x(2:4)'                          %求α β γ
```

结果是：

k=24.5085

ans=α　β　γ

ans=1.4957　-1.0000　-0.0000

上述语句用到矩阵分块应用和矩阵合并操作，请读者结合前面讲述的有关内容细细体会。

4.1.4.2　氮气热容的通用计算式

氮气的恒容摩尔热容是温度的函数，见表 4-2 的数据。

表 **4-2**　氮气的恒容摩尔热容数据

T/K	450	500	600	700	800	900	1000
$C_{V,m}/cal \cdot K^{-1}$	5.04	5.08	5.21	5.35	5.52	5.66	5.83

已知在 450K 到 1000K 的温度范围内，氮气的恒容摩尔热容可以表示为 $aT+b$，试求出 a 和 b，以给出氮气的恒容摩尔热容通用计算式。

我们有 7 组不同温度下氮气的热容数据，很自然的想法就是由这七组数据得到由 7 个方程构成的线性方程组，该线性方程组含有两个未知数 a 和 b，求解该线性方程组，得到未知数 a 和 b，从而给出氮气的恒容摩尔热容通用计算式子。这个线性方程组是矛盾线性方程组，因为方程个数大于未知数个数，我们用 MATLAB 的反除算符求解。

```
T=[450 500 600 700 800 900 1000]';                  %温度,列向量
A=[T ones(length(T),1)];                            %生成矛盾方程组的系数矩阵
C_vm=[5.04 5.08 5.21 5.35 5.52 5.66 5.83]';         %矛盾方程组右端项
X=A\ C_vm;                                          %求解矛盾方程组
a=X(1)
b=X(2)
```

运行，得到

a=0.0015

b=4.3556

因此在 450K 到 1000K 的温度范围内，氮气的恒容摩尔热容可以表示为：

$$C_{V,m}=0.0015T+4.3556$$

注意，该矛盾线性方程组的系数矩阵的第 2 列都是 1，T 和 C_vm 都是列向量，以便使用反除算符求解，因为对于 $ax=b$ 类型的线性方程组，b 是列向量，系数矩阵 a 的每一列对应某一未知数前的系数，这些系数组成列向量，构成系数矩阵 a 的每一列。

4.1.5 齐次线性方程组的通解

齐次线性方程组指右端向量是零向量的方程组，即：$AX=0$。

当这种方程组的未知数个数大于方程个数时，该齐次线性方程组就有无数解，我们现在要来看一看如何用 MATLAB 求得这样的齐次线性方程组的通解，也就是该齐次线性方程组解空间的一组基（基础解系）。

MATLAB 使用 null 函数求齐次线性方程组的通解，使用 null 函数求解这样的齐次线性方程组的具体例子如下：

求解方程组的通解：$\begin{cases} x_1+2x_2+2x_3+x_4=0 \\ 2x_1+x_2-2x_3-2x_4=0 \\ x_1-x_2-4x_3-3x_4=0 \end{cases}$

```
A=[1  2  2  1;2  1  -2  -2;1  -1  -4  -3];
format  rat     %指定有理式格式输出
B=null(A,'r')   %求解空间的有理基
```

结果是：

$$B=\begin{matrix} 2 & 5/3 \\ -2 & -4/3 \\ 1 & 0 \\ 0 & 1 \end{matrix}$$

format rat 语句功能是指定有理式格式输出而不是采用实数格式输出。在 B=null(A，'r') 语句中，A 自然是系数矩阵，参数 r 表示求解解空间的有理基，因为齐次线性方程组的基础解系也是无穷多的，这里的 r 需要用单引号（'）括起来。

下面，我们看一看用 MATLAB 求解齐次线性方程组在化学中的应用。

4.1.5.1 氧化还原方程式配平

考虑锌和稀硝酸的氧化还原反应：

$$Zn+H^{+}+NO_3^{-}\longrightarrow Zn^{2+}+NH_4^{+}+H_2O$$

要求配平该氧化还原方程式。

我们用求解齐次线性方程组的方法解决之。首先设该氧化还原反应的化学计量系数向量是：[x_1 x_2 x_3 x_4 x_5 x_6]，注意，化学计量系数对反应物为负，对生成物为正。

根据原子守恒，得出方程组：

Zn 原子守恒　$x_1+x_4=0$

H 原子守恒　$x_2+4x_5+2x_6=0$

N 原子守恒　$x_3+x_5=0$

O 原子守恒　$3x_3+x_6=0$

氧化剂得到电子数与还原剂失去的电子数应该相等：$2x_4-8x_5=0$

解上述 5 个方程构成齐次线性方程组：

```
a=[1 0 0 1 0 0;0 1 0 0 4 2;0 0 1 0 1 0;0 0 3 0 0 1;0 0 0 2 -8 0];
format  rat
B=null(a,'r');
B=B'
```

结果是

B=-4/3　　-10/3　　-1/3　　4/3　　1/3　　1

我们再整理一下，让计量系数向量的各分量都变成整数，只需要给出命令：

B=3 *B

结果是：

B=-4　　-10　　-1　　4　　1　　3

所以经过配平的氧化还原方程式是：

$$4Zn+10H^{+}+NO_3^{-}\longrightarrow 4Zn^{2+}+NH_4^{+}+3H_2O$$

4.1.5.2 复杂反应体系分析

以甲烷为原料通过变换反应制造合成气时，反应体系中包含下列组分：CO_2、H_2O、H_2、CO、CH_4、C 和 C_2H_6，确定该反应体系的独立反应数，并写出一组独立反应。

因为该反应体系共有 7 种组分，所以该反应体系中发生的任意一个化学反应的化学计量系数向量一定可以表示为：

$$v=[v_1\ v_2\ v_3\ v_4\ v_5\ v_6\ v_7]$$

向量 v 的每个分量分别是某个反应中 CO_2、H_2O、H_2、CO、CH_4、C 和 C_2H_6 的计量系数，请注意 v 向量中的分量可以取负值，取负值时表明该分量对应的是反应物，反之是生成物，如果某化合物的计量系数是 0，表明该化合物不参加该反应。

该反应体系包含 C、H、O 三种原子，该体系的任一反应发生时，在反应过程中，虽然各元素的原子可以重新组合，但每一种元素的原子数目在反应前后是不变的。

根据 C 原子守恒：$v_1+v_4+v_5+v_6+2v_7=0$

根据 H 原子守恒：$2v_2+2v_3+4v_5+6v_7=0$

根据 O 原子守恒：$2v_1+v_2+v_4=0$

解上述齐次线性方程组：

```
a=[1 0 0 1 1 1 2;0 2 2 0 4 0 6;2 1 0 1 0 0 0];
format  rat
v=null(a,'r');
v=v'
```

结果是：

$$v=\begin{matrix}-1 & 1 & -1 & 1 & 0 & 0 & 0\\ -1 & 2 & -4 & 0 & 1 & 0 & 0\\ -1 & 2 & -2 & 0 & 0 & 1 & 0\\ -2 & 4 & -7 & 0 & 0 & 0 & 1\end{matrix}$$

该齐次线性方程组的基础解系包含4个独立解向量，这四个独立解向量的线性组合能够构成该齐次线性方程组的任何一个解。所谓独立反应就是不能由其它反应经过线性组合得到的反应，而线性方程组的基础解系任何一个解亦不能通过其它解的线性组合得到，这是通过求齐次线性方程组的基础解系来求体系的一组独立反应的原因。

所以，该体系的独立反应数是4，一组独立反应是：

$$CO_2+H_2 = H_2O+CO$$

$$CO_2+4H_2 = 2H_2O+CH_4$$

$$CO_2+2H_2 = 2H_2O+C$$

$$2CO_2+7H_2 = 4H_2O+C_2H_6$$

4.2 最优化

最优化就是求函数的最大值或最小值。最优化非常重要，因为它的意义不仅仅在于求函数的最大值或最小值，还在于它能够极大地降低生产成本，获得最大经济效益。而且，最优化方法还是求解非线性方程组和进行非线性曲线拟合的有力手段。求解非线性方程组通常使用迭代法，如牛顿迭代法，但该法要求给出较好初值，才能较好的收敛，得到足够精度的解。然而，如何给出较好的初值，一直是令人头疼的问题，所以以牛顿法为代表的各种迭代方法，一直不尽如人意。而非线性方程组的求解是可以转化为求具有平方和形式的函数最小值的问题的。非线性曲线拟合实际是一最小二乘问题，最小二乘就是求平方和函数的最小值，这也是个最优化问题。因此，最优化不但有直接应用，还有间接的应用，最优化是科学计算领域中很重要的一个问题。

由于在化学化工的实践过程中遇到的函数的自变量一般都具有一定的物理意义，具有一定的范围，因此我们这章仅研究闭区域函数的最优化问题，也是函数的自变量取值具有一定范围，不存在无界的情况。

函数最优化的关键是求函数最大值点或最小值点，即函数在哪个数据点取得最大值或最小值，从这个意义上讲，求最大值的问题和求最小值的问题是可以相互转换的，因为求函数 $f(x)$ 的最小值点实际就是求函数 $-f(x)$ 的最大值点，同样，求函数 $f(x)$ 的最大值点就是求函数 $-f(x)$ 的最小值点。

4.2.1 闭区间内单峰连续函数的最小值点

闭区间单峰连续函数，就是指该连续函数在指定闭区间内仅有一个极值，自

然，这个极值要么是最大值，要么是最小值，但只有一个极值。比如正弦函数在［0 p］是单峰函数，在［0 2p］就不是单峰函数。

若函数 $f(x)$在闭区间［a b］是单峰函数且有最小值，可以用所谓的黄金搜索法找到 $f(x)$的最小值点。在讲述黄金搜索法之前，我们需要先来说一说线段的黄金分割点。一条线段有两个黄金分割点，我们分别称为第一黄金分割点和第二黄金分割点，见图 4-4：



图 4-4　线段的两个黄金分割点

上图中，线段 ab 具有两个黄金分割点 c 和 d，分别称为线段 ab 的第一黄金分割点和第二黄金分割点，我们通常说的线段的黄金分割点指的是第二黄金分割点 d。

第一黄金分割点 c 的坐标是：$x_c=x_a+(x_b-x_a)^* k_1$。

第二黄金分割点 d 的坐标是：$x_d=x_a+(x_b-x_a)^* k_2$。

其中 $k_2=\frac{\sqrt{5}-1}{2}$，$k_1=1-k_2$。

请注意，点 c 是线段 ad 的第二黄金分割点，点 d 是线段 cb 的第一黄金分割点，这两个事实很重要，是下面编写的关于黄金搜索法的 MATLAB 函数关键内容之一。

有了上面的知识准备，我们就可以来了解黄金搜索法求闭区间连续函数的最小值，当然该函数是单峰的。

若函数 $f(x)$在闭区间［a b］是单峰函数且有最小值，求其最小值点的黄金搜索法如下。

（1）取闭区间［a b］的两个黄金分割点 c、d。

（2）比较这两个黄金分割点的函数值，如果 $f(d)\geqslant f(c)$，则说明最小值点在闭区间［a d］内；否则，最小值点必然在闭区间［c b］内；无论出现上述的哪一种情况，包含最小值的闭区间被缩小了。

（3）重复步骤 1、2，不断缩小包含最小值的闭区间，直到闭区间长度$\leqslant\varepsilon$为止（ε是一个给定的很小的正数）。

（4）取包含最小值的闭区间的中点作为函数的最小值点。

黄金搜索法的 MATLAB 函数如下。

```
function    xmin=goldsearch(f,a,b,tolerance)
%黄金分割搜索求闭区间单峰连续函数最小值
%a,b:     闭区间的下限与上限,建议采用作图方法确定
% f:      字符串,定义函数
% tolerance:  允许误差,可选择的参数,默认值为 1e-6
% xmin    最小值点
        n=nargin;
```

```
if n<4
   tolerance=1e-6;
end
k2=(sqrt(5)-1)/2;k1=1-k2;
c=a+(b-a)*k1;d=a+(b-a)*k2;
while b-a>=tolerance
    if feval(f,d)>=feval(f,c)       %以下执行缩小闭区间的操作
      b=d;d=c;c=a+(b-a)*k1;   %点c是线段ad的第二黄金分割点
      else
      a=c;c=d;d=a+(b-a)*k2;   %点d是线段cb的第一黄金分割点
    end
end
xmin=(b+a)/2;
```

黄金搜索法的实质就是通过不断比较一个区间内的两个黄金分割点的函数值来不断缩小包含最小值点的闭区间。每得到一个新区间，并不需要把该新区间的两个黄金分割点的坐标都计算出来，因为该新区间的一个黄金分割点就是旧区间的其中一个黄金分割点，因此只要计算另外一个黄金分割点坐标就可以了。这就是为什么一定要选区间的两个黄金分割点函数值进行比较而不选其它两个点的原因。

4.2.1.1 ［$HC_2O_4^-$］的分布曲线的最大值点

我们从2.1.2节的图2-2草酸各种存在形式的分布曲线看出，［$HC_2O_4^-$］的分布曲线 d_1 在pH≈2.7时，存在一最大值，这个最大值点该如何精确求得呢？我们现在就利用黄金搜索法确定该最大值点。

第一步自然是定义［$HC_2O_4^-$］的分布曲线 d_1 的函数文件，但我们需要注意，黄金搜索法是求函数 $f(x)$的最小值点的，现在我们要求最大值点，所以要求函数 $f(x)$的最大值点时，应该把问题转换为求函数 $-f(x)$的最小值点。函数文件定义如下：

```
function y=f(pH)
Ka1=5.9e-2;                                  %草酸一级离解常数
Ka2=6.4e-5;                                  %草酸二级离解常数
H=10.^(- pH);                                %由pH值计算氢离子浓度
delta_1=Ka1 * H./( H.^2+Ka1 * H+Ka1 * Ka2);  %计算草酸一氢根的分布系数
y=- delta_1;                                 %求最大值问题转换为求最小值
```

由图2-2看出，函数的最值点在闭区间［1 4］内，求上述函数最小值点的MATLAB语句如下：

xmin＝goldsearch('f',1,4)

结果是：

xmin＝2.7115

因此，$[HC_2O_4^-]$ 在 pH＝2.7115 时达到最大。

4.2.1.2　基态氢原子径向分布函数的最大值点

从图2-1基态氢原子径向分布函数图看出，基态氢原子径向分布函数在闭区间[0 3]存在一最大值点，我们黄金搜索法确定该最大值点。

第一步自然也是定义基态氢原子径向分布函数的函数文件，同样需要将求最大值问题转换为求最小值。

```
function y=f(r_a0)
a0=52.9;        %Bohr 半径
Dr=4/a0 *r_a0.^2. *(exp(-r_a0)).^2;
y=- Dr;
```

求基态氢原子径向分布函数最大值点的 MATLAB 语句如下：

xmin＝goldsearch('f',0,3)

结果是：

xmin＝1.0000

因此，当 r_a0＝1 时，也就是在 Bohr 半径处，基态氢原子径向分布函数值，即核外半径电子出现的概率达到最大。

4.2.1.3　最佳反应时间

我们再看一个涉及反应动力学的求最大值点的问题。

在恒体积、等温的间歇反应器发生如下反应：

$$A+B \xrightarrow{k_1} C$$

$$B+C \xrightarrow{k_2} D$$

A、B、C、D的初始反应浓度分别用 a_0、b_0、c_0、d_0 表示。反应的各参数是 $a_0=b_0=10\text{mol/m}^3$，$c_0=d_0=0$，$k_1=k_2=0.01\text{m}^3/(\text{mol}\cdot\text{h})$。求何时C的浓度达到最大？

要解决这个问题首先要定义一个函数 f，它能够求出反应开始后任意时刻C的浓度，并且，函数 f 的输出值应该是反应开始后任意时刻C的浓度的负值，以能够使用黄金搜索法。在定义函数 f 之前，还需要定义函数 g，函数 g 定义描述上述反应动力学微分方程组的右端。

```
function p=g(t,y)
  k1=0.01; k2=0.01;        %定义反应速率常数
  p=zeros(4,1);
```

```
    %向量p的分量存储反应动力学微分方程组的右端
    p(1)=-k1 *y(1)*y(2);p(2)=k1 *y(1)*y(2)-k2 *y(2)*y(3);
    p(3)=k1 *y(1)*y(2)- k2 *y(2)*y(3);p(4)=k2 *y(2)*y(3);
function y=f(t)
%求反应开始后任意时刻C的浓度的负值
    a0=10;b0=10;c0=0;d0=0;           %各物质的初始浓度
    [ty,y]=ode45('g',[0 t],[ a0 b0 c0 d0]);  %返回各物质浓度在[0 t]变化的浓度矩阵
    y=-y(size(y,1),3);               %返回反应开始后时刻t时C的浓度的负值
```

我们先做出C的浓度的负值随时间的变化曲线，代码如下：

```
t=0.1:0.1:20;               %定义需要作图的数据点的横坐标(时间)
y=zeros(1,length(t));       %定义需要作图的数据点的纵坐标(浓度)个数
for k=1:length(t)
    y(k)=f(t(k));           %计算需要作图的数据点的纵坐标(浓度)
end
plot(t,y, 'b')              %作图
xlabel('时间/h')
ylabel('C的浓度的负值/ mol/m^3')
```

C的浓度的负值随时间的变化曲线见图4-5：

从图4-5看出，C的浓度在闭区间［6 10］有一最值点，利用黄金搜索法求该最值点的代码如下：

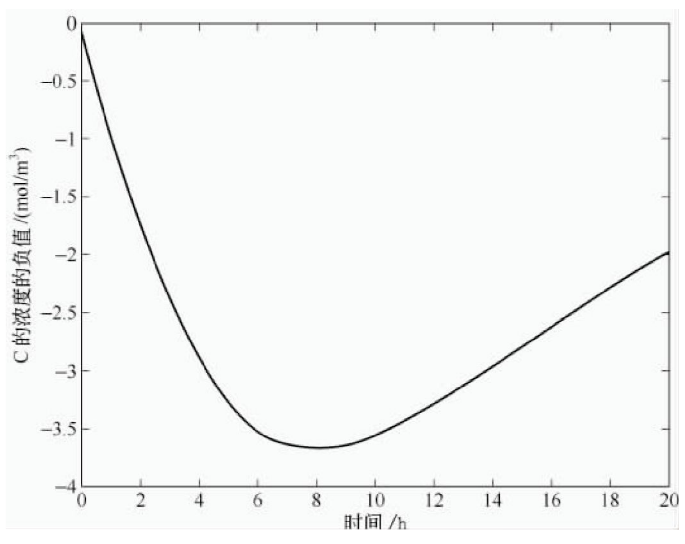


图4-5　C的浓度的负值随时间的变化曲线

```
xmin=goldsearch('f',6,10)      %利用黄金搜索法求何时C的浓度达到最大
C_max=-f(xmin)
```

结果是：

xmin=7.9696

C_max=3.6788

所以当 t=7.9696h 时，C 达到最大浓度，浓度为 3.6788mol/m³。

本问题的解决充分体现了模块化思想，最终目标是求何时 C 的浓度达到最大，需要使用黄金搜索法，使用黄金搜索法就必须知道反应开始后任意时刻 C 的浓度，反应开始后任意时刻 C 的浓度问题，是最终目标下面的第一层次的子问题，为了解决该子问题，需要解一个描述串联反应的动力学常微分方程组，这构成了第二层次的子问题，解一常微分方程组。因此，本 MATLAB 程序使用了 goldsearch 函数，然后 goldsearch 函数调用 f 函数，在 f 函数内部又调用了 ode45 函数，要求能够知道反应开始后任意时刻 C 的浓度，因此 goldsearch 函数调用了 f 函数解决这个问题，f 函数又调用了 ode45 函数，如此层层调用，最终达到最终目标。

4.2.2　函数在 n 维矩形闭区域内的最大值点

矩形闭区域边界线都是直线，对于二维空间，矩形闭区域表现为一个矩形平面，对于三维空间，矩形区域表现为一个长方体，对于四维及更高维的空间，矩形区域表现为一种超长方体。

多元函数在某一矩形闭区域的最大值问题是一个全局优化问题，就是要在整个闭区域寻找函数的最大值点。遗憾的是，过去种种最优化方法都是局部寻优的，如最速下降法、BFGS 方法和单纯型法等，这些方法都要求给定一个初值，然后从这个初值出发，通过一定的迭代步骤，最终找到离这个初值最近的极值点，注意，这里说的是极值点，而不是最终点。函数在某个闭区域的最值点只有一个，但在该闭区域的极值点可以有很多。如果运气好的话，给定的初值离最值点比较近，那么可以用这些局部寻优的方法找到最值点，但假如我们运气不好呢？所以，为了能够有百分之百把握找到真正的最值点，我们不应当使用这些局部寻优方法，我们需要使用新的全局寻优的最优化方法。

遗传算法是一种比较成功和可靠的全局寻优方法。遗传算法是基于进化论思想开发出来的一种全局寻优算法。进化论的基本思想如下：

① 通过自然选择，适应性强的个体具有更高的存活概率并繁殖下一代，适应能力较差的个体存活概率很小；

② 存活下来的个体通过自身的变异或有性生殖（遗传算法里称为交叉）繁殖新后代；

③ 新后代中可能出现适应性更强的个体。

假如我们要寻找多元函数 $f(x_1\ x_3\ x_2\cdots x_n)$在某矩形闭区域D的最大值点，我们可以随机选取闭区域D内的一批点，将这批点看做一个生物体种群，然后让这个生物体种群在闭区域内“进化”，直至该生物体种群中出现最优个体为止，该最优个体就是闭区域D的最大值点。这就是遗传算法的基本思想。根据上面的阐述，遗传算法的详细步骤如下：

① 随机产生矩形闭区域D内一批数据点，作为初始生物种群；

② 进行自然选择，保证种群中适应性强的个体具有更大的概率被选中；

③ 被选中的个体构成父代种群，发生变异或交叉，产生新一代，构成子代种群；

④ 如果子代种群的最优个体与父代种群的最优个体差异很小或者繁殖的代数达到一定数值，均停止进化，取子代种群的最优个体对应的点作为函数的最大值点；

⑤ 否则，重复步骤2、3，直到满足步骤4为止。

我们下面要看一看如何把上面的步骤转化为MATLAB程序，以实现遗传算法，在下面的叙述中，n维矩形闭区域D的数据点与函数值是和种群中的个体是相通的，是可以相互通用的。

(1) 随机产生初始种群

我们首先需要在n维矩形闭区域D内随机产生一批数据点，作为初始生物种群。这些数据点都可以用n维向量来表达，如何随机产生这些n维向量？由于向量是由各分量构成的，问题就转化为如何随机地产生各分量。更直接的问题就是如何产生一个介于两个数，如（$a\ b$），之间的随机数？

MATLAB提供函数rand，用于产生介于（0 1）之间的均匀分布随机数，但不包括0和1。一般使用格式如下：

rand(n,m)

函数rand产生一个n行m列的矩阵，该矩阵的每个元素都是介于（0 1）之间的随机数。rand函数不带参数使用时，它的作用是产生一个介于（0 1）均匀分布随机数。我们可以通过适当的运算将rand函数、a和b组合起来，产生（$a\ b$）之间的随机数。运算组合如下：

a+(b-a)*rand(n,m)

该组合产生一个n行m列的矩阵，该矩阵的每个元素都是介于（$a\ b$）之间的随机数。有了以上的讲述，我们就了解了在n维矩形闭区域D内随机产生一批数据点的核心知识。下面，我们来看一看实现这一目的的MATLAB函数。

```
function pop=initialize(num,bounds,f_name)
% 产生初始种群
% num:种群包含的个体数目
% bounds:闭区域的边界,n行2列的矩阵
```

```
% f_name:目标函数名,字符串
numVars=size(bounds,1);            %获得该闭区域的维度
rng=(bounds(:,2)-bounds(:,1))';    %闭区域上下两个边界的差值,行向量
xZomeLength=numVars+1;             %种群矩阵的列数
pop=zeros(num,xZomeLength);        %确定种群矩阵的大小
%根据闭区域的边界,随机产生一批数据点,数目为 num
pop (:,1:numVars)=(ones(num,1) * rng). * (rand(num,numVars))+(ones
(num,1) * bounds(:,1)');
for k=1:num
%获得群体中每个个体的对应的函数值
pop(k,xZomeLength)=feval(f_name,pop(k,1:numVars));
end
```

因为 n 维矩形闭区域 D 有 n 个维度，每个维度均有一定的数值范围，每个维度均需要 2 个数值来确定其下界和上界，所以 bounds 是一个 n 行 2 列矩阵，每行对应闭区域的一个维度，第一列对应每个维度的下界，第二列对应每个维度的上界。

pop 是种群矩阵，每行的第一列到倒数第二列存储随机产生的 n 维矩形闭区域 D 的一个数据点向量的各分量，最后一列存储该数据点对应的函数值，实际上该矩阵的每行也代表种群中的一个个体。种群矩阵的列数就是该种群包含的个体数目，也是随机产生的 n 维矩形闭区域 D 的数据点的个数。

在 n 维矩形闭区域 D 内随机产生一批数据点的语句使用了矩阵乘法，一次性地将这批数据点产生出来，但本质仍然是与“a+(b-a) * rand (n,m)”语句相同的。

（2）确定种群中每个个体的适应性

在随机产生矩形闭区域 D 内一批数据点，作为初始生物种群后，就应当进行自然选择步骤，保证种群中适应性强的个体具有更大的概率被选中。什么是适应强的个体呢？自然是函数值大的个体或数据点，如果能将数据点的函数值转化概率，并且函数值越大，对应的概率越大，我们称为个体的适应概率，那么就可以进行选择过程了。目前较成熟的将个体函数值转化为适应性概率的方法是采用归一化几何排序公式：

$$P=\frac{q}{1-(1-q)^{N}}(1-q)^{r-1}$$

式中　P——个体的适应性概率；

q——几何排序常数，一般取 0.08；

r——个体按从大到小排序的排序序号，函数值最大的个体排序序号为 1，概率最大；

N——种群中的个体数目。

将种群中各个体按照其函数值从大到小排序，函数值最大的个体排序序号为1，函数值最小的个体排序序号为N，N为种群中个体数目，然后将这些序号和种群中个体数目代入归一化几何排序公式，将数据点的函数值转化概率，并且函数值越大，对应的概率越大，并且这些概率之和为1。

（3）依据个体的适应性对个体进行选择

现在，我们获得了各个体的适应概率，需要进行选择。进化论认为，进化既没有目的，也没有方向，充满尝试与失败，进化具有随机性。因此，遗传算法的选择过程，也是一随机性过程。

假设种群中存在A、B、C、D，共4个个体，每个个体有不同的适应性概率，个体适应性概率的大小用矩形长度来表示，xa、xb、xc是以矩形A的左端为原点，矩形A、矩形B、矩形C右端的坐标值，见图4-6：



图4-6　种群中各个体组成的带形区域

假设我们向由矩形A、矩形B、矩形C和矩形D构成的带形领域随机投标，标扎到哪个矩形内就算哪个矩形代表的个体被选中。自然这样一种投标方式应该具有这样的特性，标扎到某个矩形的可能性的大小，仅与矩阵的长度有关，矩形越长，标扎到该矩形的可能性越大，而且这种可能性与矩形所在的位置无关。能不能用MATLAB模拟这种投标方式呢？能！MATLAB的rand函数产生的是介于（0,1）之间的均匀分布随机数，这样的随机数具有的特性就是：它们落入（0,1）区间中的某个子区间的可能性仅与该区间的长度有关，区间长度越长，落入该区间的可能性就越大，而与该子区间在（0,1）区间中的位置无关。所以我们就使用rand函数产生随机数，随机数落到哪个矩阵所在的区间，哪个矩形代表的个体就被选中，比如，若rand产生的随机数r介于（xb,xc)，那么就表现个体C被选择。

实现选择过程的MATLAB函数如下：

```
function newPop=normGeomSelect(oldPop)
% 从n个老个体中选择n个新个体
% oldPop:老种群矩阵,oldPop的行数代表个体数目,
% oldPop的最后一列储存函数值,前面各列存储数据点各分量
q=0.08;              %几何排序常数
e=size(oldPop,2);    %种群矩阵的列数
n=size(oldPop,1);    %种群个体数目
newPop=zeros(n,e);   %确定新的种群矩阵的维度,种群矩阵的最后一列存储函数值
```

```
fit=zeros(n,1);              %确定适应性概率向量的大小
x=zeros(n,2);
%x 矩阵第一列存储逆序排序号
%x 矩阵第二列存储对应一定逆序排序号的个体在原种群的位置
x(:,1)=[n:-1:1]';
[y x(:,2)]=sort(oldPop(:,e));
r=q/(1-(1-q)^n);
fit(x(:,2))=r*(1-q).^(x(:,1)-1);
                             %获得原种群各个体的适应性概率,适应性概率和为 1
fit=cumsum(fit);             %计算各个体在带形区域的范围
% 以下语句利用均匀分布随机数进行选择,
rNums=sort(rand(n,1));       %产生 n 个均匀分布随机数,准备进行 n 次选择
fitIn=1; newIn=1;
while newIn<=n
  if(rNums(newIn)<fit(fitIn))
    newPop(newIn,:)=oldPop(fitIn,:);
    newIn=newIn+1;
  else
    fitIn=fitIn + 1;
  end
end
```

normGeomSelect 选择函数中间使用函数 cumsum，该函数是 MATLAB 的固有函数，称为累计和函数。若 x 是向量，则 cumsum（x）返回与 x 相同长度的一个向量，该向量的第一个元素就是向量 x 的第一个元素，第二个元素就是向量 x 前两个元素之和，第 n 个元素就是 x 前 n 个元素的和。累计和函数用于 normGeomSelect 函数，就是为了计算出如图 4-6 中种群中各个体组成的带形区域中 xa、xb、xc 等的坐标值，以进行下一步判断均匀分布的随机数落在哪个区间，从而根据随机数落入的区间进行选择。

在用均匀分布随机数进行选择的部分，首先对均匀分布随机数进行从小到大的排序，依次先用较小的随机数进行选择，然后使用较大的随机数进行选择。这样做的好处是能够节省选择的运算量。比如某随机数落入图 4-6 的 B 区域，那么，下一次用随机数选择时，不必从 A 区域开始搜索，因为这次使用的随机数只可能落入 B 区域或 B 区域后面区域，如 C 区域。fitIn 变量就是记忆当次随机数落到哪个区域的变量，下次选择就可以从 fitIn 变量记忆的区域开始，而不必从头开始。

（4）个体的变异与交叉

在完成选择后，选择得到的种群被称为父代种群，需要进行交叉和变异，产生新的种群。在遗传算法中，交叉是指由两个父代个体经过一定变换产生两个新个体；变异指一个父代个体经过一定变换产生1个新个体。这些新个体称为子代。交叉和变异分别对应自然界的有性生殖和生物体自身的变异。

我们假设个体对应的数据点是（$x_1\ x_3\ x_2 \cdots x_i \cdots x_n$），$U(0,1)$代表介于（0,1）之间的均匀分布随机数，$U(a_i,b_i)$代表介于（$a_i$,$b_i$）之间的均匀分布随机数，$a_i$和$b_i$是矩形闭区域的第$i$个维度的下界和上界。

最常用的变异方式有均匀变异、边界变异、非均匀变异和多重非均匀变异。

① 均匀变异

$$x'_i=\begin{cases}U(a_i,b_i) & \text{如果 } i=j \\ x_i & \text{否则}\end{cases}$$

均匀变异是指先产生一个介于［1,n］之间的随机整数j，则个体数据点的第j个分量被改变，有$x_j=U(a_i,b_i)$，个体数据点的其余分量不变。

实现均匀变异MATLAB函数如下：

```
function parent=unifMutation(parent,bounds,ops)
%均匀变异函数
% parent:种群中准备变异的某个个体
% bounds:2列矩阵,第一列和第二列分别存储矩形闭区域的各维度的下界和上界
df=bounds(:,2)-bounds(:,1);
numVar=size(parent,2)-1;                          %数据点的维度
mPoint=floor(rand*numVar)+1;                      %指定数据点的变异位置
newValue=bounds(mPoint,1)+rand*df(mPoint);        %产生U(a mPoint,b mPoint)
parent(mPoint)=newValue;                          %变异
```

上面的unifMutation函数用到了floor函数，该函数是MATLAB的固有函数，功能是将小数转变为最接近该小数且小于该小数的整数。如：

```
>>floor(2.3)
ans=2
>>floor(-2.3)
ans=-3
```

unifMutation函数的最后一个输入参数ops在函数体内没有被使用，这是因为最后把所有这些函数组合起来，编写遗传算法函数时，具有很大的方便性。这种方便性，到了后面我们就会看到。

② 边界变异

$$x'_i=\begin{cases}a_i & \text{如果 } i=j \quad r<0.5 \\ b_i & \text{如果 } i=j \quad r\geqslant 0.5 \\ x_i & \text{否则}\end{cases}$$

边界变异是指先产生一个介于 $[1,n]$ 之间的随机整数和 $r=U(0,1)$，则个体数据点的第 j 个分量被改变，如果 $r<0.5$，则有 $x_j=a_j$，如果 $r\geqslant 0.5$，则 $x_j=b_j$，个体数据点的其余分量不变。

实现边界变异的 MATLAB 函数如下：

```
function parent=boundaryMutation(parent,bounds,ops)
%边界变异函数
% parent:种群中准备变异的某个个体
% bounds:2 列矩阵,第一列和第二列分别存储矩形闭区域的各维度的下界和上界
numVar=size(parent,2)-1;            %数据点的维度
mPoint=floor(rand* numVar)+1;       %指定数据点的变异位置
b=round(rand)+1;                    %若 r<0.5,则 b=1,若 r≥0.5,则 b=2
newValue=bounds(mPoint,b);          %获得变异值
parent(mPoint)=newValue;            %变异
```

边界变异函数使用了 round 函数，它是 MATLAB 的固有函数，作用是将小数转变为最接近该小数的整数，对正数，该函数就是四舍五入取整函数。如果 $r<0.5$，则有 $x_j=a_j$，如果 $r\geqslant 0.5$，则 $x_j=b_j$，这实际上是一个二元选择，利用语句 b=round(rand)+1，将二元选择转变为如果 $r<0.5$，得到整数 1，如果 $r\geqslant 0.5$，得到整数 2，再利用 newValue=bounds(mPoint,b)，获得边界变异值，因为 bounds 的第一列和第二列分别存储矩形闭区域的各维度的下界和上界。

③ 非均匀变异

$$x'_i=\begin{cases} x_i+(b_i-x_i)f(G) & \text{如果 } i=j \quad r<0.5 \\ x_i-(x_i-a_i)f(G) & \text{如果 } i=j \quad r\geqslant 0.5 \\ x_i & \text{否则} \end{cases}$$

在这里：

$$f(G)=\left[r_2\left(1-\frac{G}{G_{\max}}\right)\right]^2$$

式中　r_2——介于 $(0,1)$ 的均匀分布随机数；

G——种群的当前繁殖代数；

$G_{\max}$——遗传算法设定的最大繁殖代数；

b——形状因子，一般取为 3。

非均匀变异是指先产生一个介于 $[1,n]$ 之间的随机整数和 $r=U(0,1)$，则个体数据点的第 j 个分量被改变，如果 $r<0.5$，则有 $x_j=x_j+(b_j-x_j)$，如果 $r\geqslant 0.5$，则 $x_j=x_j-(x_j-a_j)$，个体数据点的其余分量不变。

实现非均匀变异的 MATLAB 函数如下：

```
function parent=nonUnifMutation(parent,bounds,Ops)
%非均匀变异函数
% parent:种群中准备变异的某个个体
% bounds:2 列矩阵,第一列和第二列分别存储矩形闭区域的各维度的下界和上界
% Ops:向量,存储实现非均匀变异所需要的一些参数
cg=Ops(1);                        %种群的当前繁殖代数
mg=Ops(3);                        %遗传算法设定的最大繁殖代数
b=Ops(4);                         %形状因子
df=bounds(:,2) - bounds(:,1);
numVar=size(parent,2)-1;          %数据点的维度
mPoint=floor(rand* numVar)+1;     %指定数据点的变异位置
md=round(rand);                   %md 非 0 即 1,并且取这两值是等可能的
if md
  newValue=parent(mPoint)+delta(cg,mg,bounds(mPoint,2)-parent(mPoint),b);
else
  newValue=parent(mPoint)-delta(cg,mg,parent(mPoint)-bounds(mPoint,1),b);
end
parent(mPoint)=newValue;
function change=delta(ct,mt,y,b)
%计算 f(G)
%ct:种群的当前繁殖代数
%mt:遗传算法设定的最大繁殖代数
r=ct/mt;
change=y *(rand *(1-r))^b;
```

非均匀变异函数 nonUnifMutation 用到了参数 Ops。$r<0.5$ 和 $r\geqslant 0.5$ 的二元选择结构被语句 md=round (rand) 转换为 0 和 1 的选择，md 非 0 即 1，并且取这两值是等可能的，这是很巧妙的。nonUnifMutation 函数还调用了函数 delta，进行非均匀变异新值的辅助计算。

④ 多重非均匀变异　多重非均匀变异指数据点的所有分量按照非均匀变异的计算方法发生变异。

实现多重非均匀变异的 MATLAB 函数如下：

```
function parent=multiNonUnifMutation(parent,bounds,Ops)
%多重非均匀变异函数
% parent:种群中准备变异的某个个体
```

```
% bounds:2 列矩阵,第一列和第二列分别存储矩形闭区域的各维度的下界和上界
% Ops:向量,存储实现非均匀变异所需要的一些参数
cg=Ops(1);                    %种群的当前繁殖代数
mg=Ops(3);                    %遗传算法设定的最大繁殖代数
b=Ops(4);                     %形状因子
df=bounds(:,2) - bounds(:,1);
numVar=size(parent,2)-1;      %数据点的维度
md=round(rand(1,numVar));     %md 的各分量非 0 即 1,并且取这两值是等可能的
for i=1:numVar
  if md(i)
    parent(i)=parent(i)+delta(cg,mg,bounds(i,2)-parent(i),b);
  else
    parent(i)=parent(i)-delta(cg,mg,parent(i)-bounds(i,1),b);
  end
end
```

多重非均匀变异函数 multiNonUnifMutation 用到了参数 Ops。$r<0.5$ 和 $r\geqslant 0.5$ 的二元选择结构被语句 md=round(rand(1,numVar)) 转换为 0 和 1 的选择，md 非 0 即 1，并且取这两值是等可能的，这是很巧妙的。nonUnifMutation 函数同样调用了函数 delta，进行多重非均匀变异新值的辅助计算。

最常用的交叉方式有算术交叉、简单交叉和试探式交叉。

① 算术交叉　对于两个父代个体 X、Y，产生两个新个体 X'、Y'的算术交叉的格式是：

$$X'=r*X+(1-r)*Y$$
$$Y'=(1-r)*X+r*Y$$

在这里，$r=U(0,1)$。

实现算术交叉的 MATLAB 函数如下：

```
function [c1,c2]=arithXover(p1,p2,bounds,ops)
%算术交叉交叉函数
% p1,p2:父代个体
% c1,c2:子代个体
a=rand;
c1=p1 *a + p2 *(1-a);
c2=p1 *(1-a) + p2 *a;
```

② 简单交叉　简单交叉是指首先产生一个介于 [1 n] 随机整数 j，然后遵循下

列交叉格式：

$$x'_i=\begin{cases}x_i & 如果\ i\leqslant j\\ y_i & 否则\end{cases}\qquad y'_i=\begin{cases}y_i & 如果\ i\leqslant j\\ x_i & 否则\end{cases}$$

x_i、y_i代表父代个体X、Y的分量，x_i'、y_i'代表子代个体的X'、Y'的分量。

实现简单交叉的MATLAB函数如下

```
function [c1,c2]=simpleXover(p1,p2,bounds,Ops)
%简单交叉函数
numVar=size(p1,2)-1;
cPoint=floor(rand * numVar) + 1;      %确定交叉位置
c1=[p1(1:cPoint) p2(cPoint+1:numVar+1)];
c2=[p2(1:cPoint) p1(cPoint+1:numVar+1)];
```

③ 试探式交叉　对于试探式交叉，新的子代个体使用下列规则产生：

$$X'=X+r*(X-Y)$$

$$Y'=X$$

其中$r=U(0,1)$，X是比Y适应性更强的个体，也就是个体X的函数值比个体Y的函数值要大。注意上述规则还要求得到的新个体X'的各分量不超出矩形闭区域各维的下界和上界，否则，重新产生随机数$r=U(0,1)$，再利用上述规则交叉，如果试探产生n次随机数后仍然不能满足不超出矩形闭区域各维的下界和上界的条件，则让子代个体与父代个体相同，试探次数一般可取为3次。

实现简单交叉的MATLAB函数如下

```
function [c1,c2]=heuristicXover(p1,p2,bounds,Ops)
%试探式交叉函数
% bounds:2列矩阵,第一列和第二列分别存储矩形闭区域的各维度的下界和上界
% Ops:存储试探式交叉函数需要用到的参数
retry=Ops(2);                        %试探次数
i=0;
good=0;
b1=bounds(:,1)';                     %矩形区域的下界
b2=bounds(:,2)';                     %矩形区域的上界
numVar=size(p1,2)-1;
if(p1(numVar+1) > p2(numVar+1))      %找到父代个体中适应性强的,用bt表示
   bt=p1;
   wt=p2;
else
```

```
    bt=p2;
    wt=p1;
  end
  while i<retry
    a=rand;
    c1=a *(bt - wt) + bt;
    if (c1(1:numVar) <=b2 & (c1(1:numVar) >=b1))
      i=retry;
      good=1;
    else
      i=i + 1;
    end
  end
  if(~good)
    c1=wt;
  end
  c2=bt;
```

（5）实现个体的连续进化

在写出随机产生一批数据点的函数，选择函数和各种交叉和变异的函数后，我们就需要一个主函数，将这些函数组织起来，以获得一个完成遗传算法的完整函数。自然，需要有一个遗传算法的终止条件，当达到这个条件时，停止遗传算法。一般这个条件可取种群繁殖演化的最大代数，或上一代种群中的最优个体对应的函数值与当前代最优个体对应的函数值非常接近。在这两种条件下停止遗传算法，都取当前代的最优个体作为遗传算法获得的最优解。

我们上面讲了多种变异和交叉方式，在遗传算法中用哪些呢，答案是全部用上，这样才能保证种群充分进化，找到最优解。我们一共讲了 4 种变异方式，边界变异、多重非均匀变异、非均匀变异和均匀变异各执行 4 次、6 次、4 次和 4 次。交叉方式我们讲了 3 种，这三种交叉方式各执行 2 次。

具体实现遗传算法的 MATLAB 函数如下：

```
function [x_max,y_max,n_gen]=gax(f_name,bounds,maxterm,num,tolerance)
%遗传算法求函数最大值
% f_name:M 文件名,定义要求解的函数
% bounds:2 列矩阵,第一列和第二列分别存储矩形闭区域的各维度的下界和上界
% maxterm:最大代数,可选择参数,默认值 500
```

```
% num:每代的群体数目,可选择参数,默认值80
% tolerance:误差限,可选择参数,默认值1e-6
% n_gen:运算终止时繁衍的代数
% x_max:最优点
% y_max:最大值
% n_gen:繁殖代数
m=nargin;
if m<3
    maxterm=500;
end
if m<4
    num=80;
end
if m<5
    tolerance=1e-6;
end
xOverOps=[2 0;2 3;2 0];      %规定各种交叉方式的次数和试探交叉函数的参数
mutOps=[4 0 0;6 maxterm 3;4 maxterm 3;4 0 0];
                              %规定各种变异方式的次数和有关参数
c=clock;
rand('seed',prod(c(3:6)));    %产生随机数种子
xOverFNs=['arithXover heuristicXover simpleXover'];%规定交叉方式
%规定变异方式
mutFNs=['boundaryMutation multiNonUnifMutation nonUnifMutation unifMutation'];
xOverFNs=parse(xOverFNs);
mutFNs=parse(mutFNs);
numXOvers     =size(xOverFNs,1);        %交叉的种类数
numMuts       =size(mutFNs,1);          %变异的种类数
startPop=initialize(num,bounds,f_name); %产生初始种群
gen=0;
xZomeLength   =size(startPop,2);        %种群矩阵的列数
numVar        =xZomeLength-1;           %数据点的分量数目
popSize       =num;                     %种群中个体数
while 1
    endPop=normGeomSelect(startPop);    %选择
    %进行交叉操作
```

```
for i=1:numXOvers
   for j=1:xOverOps(i,1),
        a=floor(rand*popSize)+1;                    %确定交叉个体
        b=floor(rand*popSize)+1;                    %确定交叉个体
        xN=deblank(xOverFNs(i,:));                  %选择交叉函数
        [c1,c2]=feval(xN,endPop(a,:),endPop(b,:),bounds,xOverOps(i,:));
        c1(xZomeLength)=feval(f_name,c1(1:numVar));
        c2(xZomeLength)=feval(f_name,c2(1:numVar));
        endPop(a,:)=c1;                             %交叉后的新个体
        endPop(b,:)=c2;                             %交叉后的新个体
   end
 end
 %进行变异操作
 for i=1:numMuts
   for j=1:mutOps(i,1),
      a=floor(rand*popSize)+1;                      %确定变异个体
      c1=feval(deblank(mutFNs(i,:)),endPop(a,:),bounds,[gen mutOps(i,:)]);
      c1(xZomeLength)=feval(f_name,c1(1:numVar));
      endPop(a,:)=c1;                               %变异后的新个体
   end
 end
 gen=gen+1;
 [y2,row2]=max(endPop(:,xZomeLength));   %当前代的最优个体
 [y1,row1]=min(startPop(:,xZomeLength)); %上一代的最优个体
 %终止终止遗传算法的条件:
 %上一代种群中的最优个体的函数值与当前代最优个体的函数值非常接近
 if abs(y2-y1)<tolerance
     x_max=endPop(row2,1:numVar);                   %最优点
     y_max=y2;                                      %最大函数值
     n_gen=gen;                                     %繁殖代数
     return
end
startPop=endPop;
%终止终止遗传算法的条件:
%种群繁殖演化到规定的最大代数
if gen>=maxterm
```

```
        x_max=endPop(row2,1:numVar);            %最优点
        y_max=y2;                               %最大函数值
        n_gen=gen;                              %繁殖代数
        return
    end
end
```

gax函数中使用了c=clock和rand('seed',prod(c(3:6)))语句，clock是MATLAB的固有函数，其作用是产生向量，其分量依次是当前的，以实数表示的年、月、日、时、分和秒。prod也是MATLAB的固有函数，当prod函数的自变量是一向量时，prod返回向量的各分量的积。由于使用MATLAB的rand函数产生随机数时，一般要给出一个数作为随机数种子，以这个种子作为起点，依据一定规则，产生随机数序列，rand('seed',prod(c(3:6)))就是以prod(c(3:6))作为随机数种子，来产生随机数序列，由于prod(c(3:6))每时每刻都在变动，所以使用时间之积作为种子，能够保证随机数序列最大的随机性。

在gax函数的交叉和变异部分的语句，使用了产生随机整数的语句，这是因为需要从父代群体中确定个体进行交叉和变异，这里使用随机数整数确定进行交叉和变异的个体。

gax函数使用了deblank函数，它是MATLAB的固有函数，作用是去除字符串的尾部空格。

gax函数调用了函数parse，该函数的代码及其说明如下：

```
function x=parse(inStr)
% 将一行排开的、用空格分隔的字符串变成按一列排列、左对齐的字符序列
%' This is an apple' 将变成
%   'This          '
%   'is            '
%   'an            '
%   'apple         '
inStr=fliplr(deblank(fliplr(inStr)));
strLen=length(inStr);
max1=strLen;
x=[];
while sign(strLen)==1
        [a,inStr,strLen]=parsex(inStr,max1);
        x=[x;a];
```

```
end
parse函数又调用如下函数parsex:
function [a,inStr,strLen]=parsex(inStr,max1)
        l=1;
        while 1
           if l<=length(inStr) & inStr(l)~=''
              l=l+1;
           else
              a=[inStr(1:l-1),blanks(max1-l+1)];
              inStr=fliplr(deblank(fliplr(inStr(l:length(inStr)))));
              strLen=length(inStr);
              return
           end
        end
```

parse和parsex调用了MATLAB的固有函数fliplr，fliplr将原向量颠倒排列，返回一个颠倒排列的向量。如：

```
>>a=[1 3 4 6];
>>fliplr(a)
```

ans=6　4　3　1

将gax、initialize、boundaryMutation、arithXover等都保存在MATLAB的安装目录的子目录work下，就可以使用遗传算法求矩形闭区域函数的最大值了。

作为gax使用的例子，我们求如下函数在矩形闭区域闭区域［-10 10;-10 10］的最大值。

$$z=xe^{-x^2-y^2}$$

实现需要编写函数文件：

```
function y=f(x)
        y =x(1)*exp(-x(1)^2-x(2)^2);
```

x(1)代表x，x(2)代表y。

求 $z=xe^{-x^2-y^2}$ 在矩形闭区域闭区域［-10 10;-10 10］最大值的MATLAB的代码如下：

```
>>[x_max,y_max,n_gen]=gax('f',[-10 10;-10 10])
x _max=0.7071      0.0000
y _max=0.4289
```

n _gen=500

因此，经过种群繁衍 500 代，求得函数 $z=xe^{-x^2-y^2}$ 在矩形闭区域闭区域 [-10 10;-10 10] 的点 (0.7071,0.0000) 取得最大值，最大值为 0.4289。

遗传算法是一种随机性的全局优化方法，如果能将遗传算法和传统的局部优化算法（如单纯形法、BFGS 法、最速下降法等）结合起来，将能收到更好的效果。先由遗传算法找到大概的最大值点，然后以这个大概的最大值点为局部优化算法的迭代初值，进一步精确确定最大值点。

具体使用格式是：遗传算法+局部优化方法。

最常用的局部优化算法是单纯形方法，MATLAB 用 fminsearch 函数实现，fminsearch 函数的使用格式是：

[x,fval]=fminsearch(fun,x0)

fun 可以是函数句柄，也可以是代表函数文件名的字符串。x_0 是迭代初值。x 是函数 fun 的最小值点，feval 是 fun 函数的最小值。

这些局部优化方法都是非常成熟的方法，MATLAB 本身就提供这些局部优化函数，这些函数的详细使用方法请读者参考 MATLAB 的有关帮助文档或手册。

4.2.3 平方和形式的函数的最小值点

我们接下来讨论一类函数的最小值问题，这类函数具有平方和的形式：

$F(x_1\ x_2\ x_3\cdots x_n)=f_1(x_1\ x_2\ x_3\cdots x_n)^2+\cdots+f_{m-1}(x_1\ x_2\ x_3\cdots x_n)^2+f_m(x_1\ x_2\ x_3\cdots x_n)^2$

解决此类函数的最小值问题，具有极大的意义！因为求非线性方程组的根的问题和最小二乘问题，都可以转化为求上述函数最小值点的问题。因此，我们下面就来讨论如何求解这类函数的最小值点。

我们将函数 $F(x_1\ x_2 x_3\cdots x_n)$ 用求和符号改写为：

$$F=\sum_{i=1}^{m}f_i(x_1,x_2,\cdots,x_n)^2$$

假设 $x'_1x'_2\cdots x'_n$ 和 $x_1x_2x_3\cdots x_n$ 均是非常接近函数 F 的最小值点，所以：

$$F_{\min}\approx\sum_{i=1}^{m}f_i(x_1+(x'_1-x_1),x_2+(x'_2-x_2),\cdots,x_n+(x'_n-x_n))^2$$

将函数 $f_i(i=1\cdots m)$ 在 $x_1x_2x_3\cdots x_n$ 进行泰勒展开，忽略展开后的二次项和更高次项，得到：

$$F_{\min}\approx\sum_{i=1}^{m}[f_i(x_1,x_2,\cdots,x_n)+\sum_{k=1}^{n}\frac{\partial f_i}{\partial x_k}x_1+(x'_k-x_k)]^2$$

为了弥补忽略展开后 Δ 的二次项和更高次项造成的误差，引入阻尼因子 d，如下：

$$F_{\min}\approx\sum_{i=1}^{m}[f_i(x_1,x_2,\cdots,x_n)+\sum_{k=1}^{n}\frac{\partial f_i}{\partial x_k}(x'_k-x_k)]^2+d\sum_{k=1}^{n}(x'_k-x_k)^2$$

我们设：

$$Q=\sum_{i=1}^{m}[f_i(x_1,x_2,\cdots x_n)+\sum_{k=1}^{n}\frac{\partial f_i}{\partial x_k}(x'_k-x_k)]^2+d\sum_{k=1}^{n}(x'_k-x_k)^2$$

如果选择合适的 d 值并求出使得 Q 达到最小的 $x'_1x'_2\cdots x_n$ 值，并使得 F 更小，那么我们就得到更接近 F 最小值点的新近似值 $x'_1x'_2\cdots x'_n$。

我们令 $(x_1x_2x_3\cdots x_n)=(x'_1x'_2x'_3\cdots x'_n)$，然后代入 Q，再求得新的 $x'_1x'_2\cdots x'_n$。如此反复进行上述操作，直到 $|F'-F|\leqslant\varepsilon$ 为止（ε 是一个给定的很小的正数）。这时取最新的（$x'_1\ x'_2\ x'_3\cdots x'_n$）为函数 F 的最小值点。现在问题的关键就转化为如何求出 $x'_1x'_2\cdots x'_n$ 值，使 Q 达到最小。这个问题，实际就是求解如下矛盾线性方程组：

$$\sum_{k=1}^{n}\frac{\partial f_i}{\partial x_k}x'_k=-f_i(x_1,x_2,\cdots x_n)+\sum_{k=1}^{n}\frac{\partial f_i}{\partial x_k}x_k \qquad (i=1,2,3\cdots m)$$

$$\sqrt{d}x'_k=\sqrt{d}x_k \qquad (k=1,2,3\cdots n)$$

共有 $m+n$ 个线性方程，未知数 n 个，可以使用 MATLAB 的反除算符（\）求解。

现在的问题就剩下如何确定阻尼因子 d，为了减少迭代次数，d 宜选用较小的值，仅当不能保证相应的 Q 值比前次小的情况下，才选用较大的 d 值，因此阻尼因子的确定规则如下：

① 指定常数 $c=10$ 和 $d_0=0.01$；

② 算出初值 $x_1\ x_2\ x_3\cdots x_n$ 对应的 F；

③ 设 $a=-1$；

④ 令 $d=c^a*d_0$；

⑤ 根据 d 值，得到新的 $x'_1x'_2\cdots x'_n$，进而得到新的 F' 值；

⑥ 如果 $F>F'$，那么令（$x_1\ x_2\ x_3\cdots x_n$）=（$x'_1\ x'_2\ x'_3\cdots x'_n$），$F=F'$，$d_0=d$，返回 3，直到满足迭代终止条件；

否则 $a=a+1$，返回 4。

求具有平方和形式的函数 F 最小值点的 MATLAB 代码是：

```
function [x_min,y_min]=marquet_F2(f_name,x0,tolerance,max1)
% 麦夸脱方法平方和形式的函数的最小值
% f_name:字符串变量,函数向量,由函数 M 文件定义
% 定义格式:f(x),y=[f1;f2;…fn]或[f1,f2,…fn]
% x0:初始点坐标
% tolerance:误差限,默认值 1e-6
% max1:最大循环次数,可选择参数,默认值为 500
% x_min:极小值点
% y_min:极小点函数值
        m=nargin;
        if m<3
```

```
        tolerance=1e-6;
    end
    if m<4
        max1=1000;
    end
    c=10;alpha=-1;d0=0.01;          %规定阻尼因子的初值和其它条件
    if size(x0,1)==1                %保证解向量是列向量
        x0=x0';
    end
    k=0;                            %记录迭代次数
    f0=feval(f_name,x0);            %保证函数向量是列向量
    if size(f0,1)==1
      f0=f0';
    end
    F0=dot(f0,f0);                  %求 F 值
    while 1
        a=matrix_jocbi(f_name,x0);  %求函数向量的雅可比矩阵
        b=a*x0-f0;
        while 1
            d=c^alpha*d0;           %确定阻尼因子
            A=[a;diag(sqrt(d)*ones(1,length(x0)))];
            B=[b;sqrt(d)*x0];
            x=A\B;                  %解矛盾线性方程组,求新的解向量
            f=feval(f_name,x);
            if size(f,1)==1
              f=f';
            end
            F=dot(f,f);
            if abs(F-F0)<tolerance  %满足终止迭代条件
                x_min=x;y_min=F;
                return
            end
            if F<F0                 %得到更小的 F 值
              x0=x;d0=d;F0=F; f0=f;alpha=-1;k=0;
              break
            else
```

```
            alpha=alpha+1;
          end
          if k>max1                % 不满足终止迭代条件
            x_min=NaN;y_min=NaN;
            return
          end
          k=k+1;                   % 计算迭代次数
      end
    end
```

4.3　非线性方程组的求解

在化学计算过程中，我们会遇到求解非线性方程组的问题，如涉及多个独立反应的复杂反应体系的化学平衡计算、溶液 pH 值的计算都可归结为求解非线性方程组问题。

所谓求解非线性方程组问题，就是求一组 $[x_1\ x_2\ x_3 \cdots x_n]$，使得：

$$f_1(x_1\ x_2 \cdots x_n)=0$$

$$f_2(x_1\ x_2 \cdots x_n)=0$$

$$\vdots$$

$$f_n(x_1\ x_2 \cdots x_n)=0$$

其中 f_1、$f_2 \cdots f_n$ 是非线性函数。

迄今为止，前人已经开发出多种求解非线性方程组的方法，但仍然没有一种成熟的方法可以求解各种各样的非线性方程组。目前求解非线性方程组比较好的方法是牛顿-麦夸脱方法和维格斯坦加速方法，但这两种方法事先需要一组初始值，然后从这组初始值出发，通过反复迭代，获得非线性方程组的解（图 4-7）。自然，提供的初始值越接近非线性方程组的解，这两种方法就越有可能通过迭代获得非线性方程组高度精确的解。那么，如何使得提供的初始值尽量接近非线性方程组的解呢？

注意到求解上述非线性方程组的问题等价于下列最优化问题：求具有平方和形式的函数 $F(x_1\ x_2\ x_3 \cdots x_n)$ 的最小值点的问题。

$$F(x_1\ x_2\ x_3 \cdots x_n)=f_1(x_1\ x_2\ x_3 \cdots x_n)^2+f_2(x_1\ x_2\ x_3 \cdots x_n)^2+\cdots+f_n(x_1\ x_2\ x_3 \cdots x_n)^2$$

我们可以采用具有全局寻优能力的遗传算法，求得具有平方和形式的函数 $F(x_1\ x_2\ x_3 \cdots x_n)$ 的最小值点，以此最小值点为牛顿-麦夸脱方法或维格斯坦加速方法的初始值。因此，求解非线性方程组的方法可以归结为：

我们以前讲述过遗传算法的原理及其 MATLAB 函数 gax，需要注意的是 gax 函数是求函数的最大值点，因此用到求函数 $F(x_1\ x_2\ x_3 \cdots x_n)$ 最小值点时，应该用

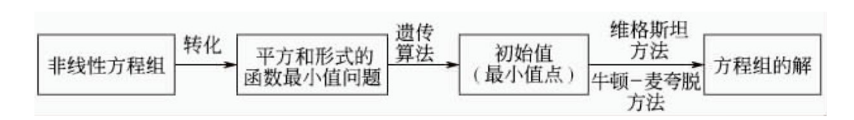


图 4-7　求解非线性方程组的方法

gax 去求函数 $-F(x_1\ x_2\ x_3\cdots x_n)$ 的最大值点，该最大值点就是函数 $F(x_1\ x_2\ x_3\cdots x_n)$ 的最小值点，也是牛顿-麦夸脱方法或维格斯坦加速方法的初值。

我们上一章讲述过牛顿-麦夸脱方法的基本原理，对于求解非线性方程组，我们把迭代终止条件定义为：$F(x_1\ x_2\ x_3\cdots x_n)\leqslant\varepsilon$（ε 是一个给定的很小的正数）。这样的迭代终止条件和我们前面讲述牛顿-麦夸脱方法的迭代终止条件有所不同，对于求一般的平方和形式的函数的最小值，我们并不知道这个最小值是什么，所以我们用 $|F'-F|\leqslant\varepsilon$（ε 是一个给定的很小的正数）作为迭代终止条件。但我们在求解非线性方程组时，我们其实已经预先知道函数 $F(x_1\ x_2\ x_3\cdots x_n)$ 的最小值必定是 0，因此 $F(x_1\ x_2\ x_3\cdots x_n)\leqslant\varepsilon$（ε 是一个给定的很小的正数）是更好的迭代终止条件。除了迭代终止条件，牛顿-麦夸脱方法的其它部分不变。

实现牛顿-麦夸脱方法求解非线性方程组的 MATLAB 函数如下：

```
function [x_min,y_min]=marquet_f0(f_name,x0,tolerance,max1)
% 麦夸脱方法求解非线性方程组
% f_name:字符串变量,函数向量,由函数 M 文件定义
% 定义格式:f(x),y=[f1;f2;…;fn]或[f1,f2,…,fn]
% x0:初始点坐标
% tolerance:误差限,默认值 1e-6
% max1:最大循环次数,可选择参数,默认值为 500
% x_min:极小值点
% y_min:极小点函数值
        m=nargin;
        if m<3
            tolerance=1e-6;
        end
        if m<4
            max1=1000;
        end
        c=10;alpha=-1;d0=0.01;   %规定阻尼因子的初值和其它条件
        if size(x0,1)==1           %保证解向量是列向量
            x0=x0';
        end
```

```
k=0;                                %记录迭代次数
f0=feval(f_name,x0);                %保证函数向量是列向量
if size(f0,1)==1
  f0=f0';
end
F0=dot(f0,f0);                      %求 F 值
while 1
    a=matrix_jocbi(f_name,x0);      %求函数向量的雅可比矩阵
    b=a * x0-f0;
    while 1
        d=c^alpha * d0;             %确定阻尼因子
        A=[a;diag(sqrt(d) * ones(1,length(x0)))];
        B=[b;sqrt(d) * x0];
        x=A\B;                      %解矛盾线性方程组,求新的解向量
        f=feval(f_name,x);
        if size(f,1)==1
          f=f';
        end
        F=dot(f,f);
        if F<tolerance              %满足终止迭代条件
            x_min=x;y_min=F;
            return
        end
        if F<F0                     %得到更小的 F 值
            x0=x;d0=d;F0=F; f0=f;alpha=-1;k=0;
            break
        else
            alpha=alpha+1;
        end
        if k>max1                   %不满足终止迭代条件
            x_min=NaN;y_min=NaN;
            return
        end
        k=k+1;                      %计算迭代次数
    end
  end
```

marquet _f0函数中用到了MATLAB的固有函数diag，对于向量x，函数diag(x)的作用在于生成一$n\times n$对角阵，其中n是向量x的长度。marquet _f0函数比较长，其实marquet _f0函数包括了两个迭代，一个是确定适合阻尼因子d的迭代，一个是确定非线性方程组解x的迭代，请读者结合以前所讲的算法细细体会，掌握将算法转变为程序的技巧与方法。

在求解非线性方程的部分，我们讲述了利用维格斯坦加速方法求解非线性方程，对于非线性方程组：

$$x_1=g_1(x_1\ x_2\cdots x_n)$$

$$x_2=g_2(x_1\ x_2\cdots x_n)$$

$$\vdots$$

$$x_n=g_n(x_1\ x_2\cdots x_n)$$

同样可以利用维格斯坦加速方法求解。设定初始值（$x_1\ x_2\ x_3\cdots x_n)^{(0)}$，用不动点迭代法：

$$x_1^{(1)}=g_1(x_1\ x_2\cdots x_n)^0$$

$$x_2^{(1)}=g_2(x_1\ x_2\cdots x_n)^0$$

$$\vdots$$

$$x_n^{(1)}=g_n(x_1\ x_2\cdots x_n)^0$$

求迭代系列的第一个点（$x_1\ x_2\ x_3\cdots x_n)^{(1)}$，并在以后的迭代采用以下格式：

$$x_i^{(p+1)}=t_ig_i(x_1^{(p)},x_2^{(p)},\cdots,x_n^{(p)})+(1-t_i)x_i^{(p)}\qquad i=1,2,\cdots,n$$

其中：

$$t_i=\frac{1}{1-s_i}=\frac{1}{1-\dfrac{g_i(x_1^{(p)},x_2^{(p)},\cdots,x_n^{(p)})-g_i(x_1^{(p-1)},x_2^{(p-1)},\cdots,x_n^{(p-1)})}{x_i^{(p)}-x_i^{(p-1)}}}\quad p=1,2,\cdots$$

当

$$\max[\,|x_1^{(p+1)}-g_1(x_1^{(p)},x_2^{(p)},\cdots,x_n^{(p)})|,|x_2^{(p+1)}-g_2(x_1^{(p)},x_2^{(p)},\cdots,x_n^{(p)})|\cdots |x_n^{(p+1)}-g_n(x_1^{(p)},x_2^{(p)},\cdots,x_n^{(p)})|\,]$$

小于ε（ε是一个给定的很小的正数）时，取（$x_1^{(p+1)},x_2^{(p+1)},\cdots,x_n^{(p+1)}$）作为非线性方程的解。

事实上，只需要把以前的维格斯坦加速方法的函数Wegstein _root中的乘除运算符改为点乘、点除算符，收敛条件改为最大范数形式，就可以将该函数用于求解非线性方程组。

所谓向量的范数，是向量的一种度量，向量范数包括1-范数、2-范数和最大范数，向量$X=(x_1\ x_2\ x_3\cdots x_n)$的1-范数为$\sum_{i=1}^{n}|x_i|$，2-范数为$\sqrt{\sum_{i=1}^{n}|x_i|^2}$最大范数为$\max_{1\leqslant i\leqslant n}|x_i|$。

我们在下面的 Wegstein _root _xy 函数中，用到了求向量 x 的最大范数的 MATLAB 函数：

```
norm(x,inf)
function root=Wegstein_root_xy(g_name,x0,tolerance,m)
%      维格斯坦加速方法求非线性方程组的解,如迭代不收敛,返回 NaN
%      g_name:字符串,函数名,也是函数文件名,定义函数 g(x)
%      x0 :迭代初值
%      tolerance :允许误差,可选择的参数,默认值 1e-6
%      m:最大迭代次数,默认值 200
n=nargin;                                          %确定函数自变量个数
if n<3
    tolerance=1e-6;                                %默认迭代精度
end
if n<4
    m=200;                                         %默认迭代次数
end
x1=feval(g_name,x0);                               %计算 g(x0)
k=0;
if     norm(abs(x1-x0),inf)>tolerance              %设定迭代条件 1
        while 1
          s=(feval(g_name,x1)-feval(g_name,x0))./(x1-x0);
          t=1./(1-s);                              %以上语句确定权值 t
          x2=t.*feval(g_name,x1)+(1-t).*x1;        %产生新的迭代初值
          if norm(abs(feval(g_name,x2)-x2),inf)<tolerance %设定迭代条件 2
              break                                %如符合迭代条件 2,
                                                    跳出循环
          else
              x0=x1;
              x1=x2;
          end
          k=k+1;
          if k>m
                root=NaN;
                return
             end
        end;
```

```
else
        root=(x0+x1)/2;         %符合迭代条件1,返回方程的根
        return
end
root=x2;                        %符合迭代条件2,返回方程的根
```

4.3.1 复杂反应体系的化学平衡计算

反应体系的化学平衡计算可看成一类特殊的物料衡算问题，即体系的终了状态是化学平衡状态时的物料衡算问题。对复杂反应体系进行物料衡算时，已知一组独立反应中各独立反应的反应进度后，即可根据化学计量关系求得各组分的反应量。因此，在求解复杂反应体系的化学平衡问题时，首先应该通过化学计量学分析确定一组独立反应，然后利用体系的终了状态是化学平衡状态这一约束条件确定各独立反应的反应进度，即可根据化学计量关系求得各组分的反应量，最终获得体系的平衡组成。

甲烷水蒸气转化制合成气可用下列两独立反应描述反应过程中的组成变化：

$$CH_4+H_2O \longrightarrow 3H_2+CO \qquad (1)$$

$$CO+H_2O \longrightarrow H_2+CO_2 \qquad (2)$$

计算在1000K时，1.2MPa，水蒸气/甲烷进料摩尔比为6的条件下该反应体系的化学平衡组成。

已知1000K时，上述两反应的化学平衡常数分别为0.26722MPa^2和1.368。以1mol甲烷为基准，设达到化学平衡时反应（1）的反应进度是$x1$，反应（2）的反应进度是$x2$，根据化学计量方程可写出化学平衡时各组分的物质的量：

CH_4	$1-x1$
H_2O	$6-x1-x2$
H_2	$3x1+x2$
CO	$x1-x2$
CO_2	$x2$
总物质的量：	$7+2x1$

于是，可得化学平衡方程：

$$\frac{(3x1+x2)^3(x1-x2)1.2^2}{(1-x1)(6-x1-x2)(7+2x1)^2}=0.26722 \qquad (1)$$

$$\frac{(3x1+x2)x2}{(x1-x2)(6-x1-x2)}=1.368 \qquad (2)$$

根据反应进度定义，有：$0<x_1<1$，$0<x_2<6$

要用MATLAB求解上述非线性方程组，首先定义函数向量文件：

```
function y=f(x)
y(1)=(3 *x(1)+x(2))^3 *(x(1)-x(2))*1.2^2-0.26722 *(1-x(1))*
(6-x(1)-x(2))*(7+2 *x(1))^2;
y(2)=(3 *x(1)+x(2))*x(2)- 1.368 *(x(1)-x(2))*( 6-x(1)-x(2));
```

利用牛顿-麦夸脱方法求解上述非线性方程组的 MATLAB 代码：

```
h=@(x)-dot(f(x),f(x));                %定义函数-F
[x_max,y_max,n_gen]=gax(h,[0 1;0 6]); %利用遗传算法求迭代初值
[x_min,y_min]=marquet_f0('f',x_max)   %利用麦夸脱方法求方程组的解
x=x_min;
total=7+2 *x(1);                      %以下语句求平衡组成,以 1mol 甲烷为基准
CH4=(1-x(1))/ total
H2O=(6-x(1)-x(2)) / total
H2=(3 *x(1)+x(2)) / total
CO=(x(1)-x(2)) / total
CO2=x(2) / total
```

计算结果是：

```
x _min=0.8600   0.5717
y _min=4.1137e-014
CH4=0.0161
H2O=0.5239
H2=0.3614
CO=0.0331
CO2=0.0656
```

注意上述代码中，通过函数句柄定义了 $-F$，利用函数 gax 求 $-F$ 的最大值点，也是函数 F 的最小值点。

利用维格斯坦加速方法求解上述非线性方程组的 MATLAB 代码如下：

```
h=@(x)-dot(f(x),f(x));                 %定义函数-F
[x_max,y_max,n_gen]=gax(h,[0 1;0 6]);
g=@(x)x+f(x);                          %定义函数 g
x_min= Wegstein_root_xy (g,x_max)
y_min=-h(x_min)
x=x_min;
total=7+2 *x(1);
```

```
CH4=(1-x(1))/ total
H2O=(6-x(1)-x(2)) / total
H2=(3 *x(1)+x(2)) / total
CO=(x(1)-x(2)) / total
CO2=x(2) / total
```

计算结果是：

```
x _min=
    0.8600
    0.5717
y _min=5.1892e-014
CH4=0.0161
H2O=0.5239
H2=0.3614
CO=0.0331
CO2=0.0656
```

值得注意的是上述代码用函数句柄定义了函数 $g(x)$，然后利用维格斯坦加速方法求得非线性方程组的解。

4.3.2 H_3PO_4 溶液中各种离子浓度的大小

计算 $0.15mol\cdot L^{-1}$ H_3PO_4 溶液中各种离子浓度的大小。

$K_1^\ominus=7.1\times10^{-3}$

$K_2^\ominus=6.3\times10^{-8}$

$K_3^\ominus=4.2\times10^{-13}$

H_3PO_4 会在水中发生电离并达到平衡，产生各种磷酸根离子、氢离子，因此问题的关键是根据磷酸的电离平衡计算出各种离子的浓度。

磷酸的三级离解反应是：

$$H_3PO_4 = H^+ + H_2PO_4^-$$

$$H_2PO_4^- = H^+ + HPO_4^{2-}$$

$$HPO_4^{2-} = H^+ + PO_4^{3-}$$

考虑到水的电离：$H_2O = H^+ + OH^-$

设三级电离反应的反应进度分别是 x_1、x_2 和 x_3，水电离的反应进度为 x_4，那么达到电离平衡后，各离子的浓度（$mol\cdot L^{-1}$）为：

H_3PO_4	$0.15-x_1$
H^+	$x_1+x_2+x_3+x_4$
$H_2PO_4^-$	x_1-x_2
HPO_4^{2-}	x_2-x_3

PO_4^{3-}　　　　　　　　x_3

OH^-　　　　　　　　x_4

由此获得4个平衡方程：

$$\frac{(x_1+x_2+x_3+x_4)(x_1-x_2)}{0.15-x_1}=7.1\times10^{-3}$$

$$\frac{(x_1+x_2+x_3+x_4)(x_2-x_3)}{x_1-x_2}=6.3\times10^{-8}$$

$$\frac{(x_1+x_2+x_3+x_4)x_3}{x_2-x_3}=4.2\times10^{-13}$$

$$(x_1+x_2+x_3+x_4)x_4=10^{-14}$$

根据化学计量学知识，$0<x_1$，x_2，$x_3<0.15$，$0<x_4<0.45$。

要用MATLAB求解上述非线性方程组，首先定义函数向量文件：

```
function y=f(x)
y(1)=1/(7.1e-3)*sum(x)*(x(1)-x(2))- (0.15-x(1));
y(2)=1/(6.3e-8)*sum(x)*(x(2)-x(3))- (x(1)-x(2));
y(3)=1/(4.2e-13)*sum(x)*x(3)- (x(2)-x(3));
y(4)=1/(1e-14)*sum(x)*x(4)-1;
```

求各离子浓度的MATLAB代码：

```
format long
h=@(x)-dot(f(x),f(x));                    %定义函数-F
[x_max,y_max,n_gen]=gax(h,[0 0.15;0 0.15;0 0.15; 0 0.45]);
                                           %利用遗传算法求迭代初值
[x_min,y_min]=marquet_f0('f',x_max)  %利用麦夸脱方法求方程组的解
x=x_min;
H3PO4=0.15-x(1)
H=sum(x)
H2PO4=x(1)-x(2)
HPO4=x(2)-x(3)
PO4=x(3)
OH=x(4)
```

结果如下：

```
        0.02927685638304
        0.00000006299973
x _min=
        0.00000000000000
        0.00000000000034
```

y _min=3.083520351763214e-015

H3PO4=0.12072314361696

H=0.02927691938311

H2PO4=0.02927679338331

HPO4=6.299972886535582e-008

PO4=9.037797742971667e-019

OH=3.415660105640587e-013

求得的结果如上，为了完整显示结果，代码开头使用了 format long 语句，将各数据用 long 格式显示。

利用维格斯坦加速方法求解上述非线性方程组的 MATLAB 代码如下：

```
format long
h=@(x)-dot(f(x),f(x));                    %定义函数-F
[x_max,y_max,n_gen]=gax(h,[0 0.15;0 0.15;0 0.15; 0 0.45]);
%利用遗传算法求迭代初值
g=@(x)x+f(x);                             %定义函数 g
x_min= Wegstein_root_xy (g,x_max)
y_min=-h(x_min)
x=x_min;
H3PO4=0.15-x(1)
H=sum(x)
H2PO4=x(1)-x(2)
HPO4=x(2)-x(3)
PO4=x(3)
OH=x(4)
```

计算结果同上。

```
         0.02927685638304
         0.00000006299973
x _min=
         0.00000000000000
         0.00000000000034
```

y _min=1.022619142394714e-016

H3PO4=0.12072314361696

H=0.02927691938311

H2PO4=0.02927679338331

HPO4=6.299972886535415e-008

PO4=9.037797227571682e-019

OH＝3.415659881429676e－013

在定义函数向量 f 时，一般要把函数向量的各函数（平衡方程）改写成多项式形式而不写成原来的分式形式，以避免某些组分的摩尔分数非常小的时候可能出现的除数为零的困难，并消除除法运算带来的非线性。此外，如果平衡常数 $K \ll 1$，一般要求平衡常数 K 以 $1/K$ 的形式出现在函数向量的各函数中，如我们上面求磷酸溶液中各离子浓度而定义的函数向量中，各平衡方程的平衡常数极小，因此出现在函数向量中的平衡常数都以倒数形式出现。

第5章
数据处理

将实验中获得的原始数据（通常是一系列离散数据）通过某种变换手段，将其转化为新的数据，获得从原始数据不能得到的新的信息，称为数据处理。

我们前面讲述的根据实验数据作图，可以看作数据处理的一种手段。此外数据处理还包括利用已有的实验数据估计出未知的实验数据、求由离散实验数据点表达的函数的微分和积分以及根据实验数据拟合出直线或曲线，都属于数据处理。这些数据处理用到的变换手段分别是利用插值、数值微分与积分和最小二乘法。数据处理还包括利用统计学的方法进行数据处理，如求数据点的平均值、标准差、真实值的置信区间和利用假设检验判断系统误差、比较不同工艺的优劣等。

5.1 插值问题的提法

二氧化硫20℃的溶解度数据见表5-1。

表5-1 二氧化硫20℃的溶解度数据

二氧化硫/(kg SO_2/100kg H_2O)	1	0.7	0.5	0.3	0.15
液面上 SO_2 分压/mmHg	59	39	26	14.7	5.8

现在我们想知道20℃时，当液面上 SO_2 分压为8.4mmHg（1mmHg=133.32Pa）时，二氧化硫的溶解度数据。我们当然可以采用实验方法，测定该分压处的二氧化硫的溶解度。但这样做的缺点是增加了额外的工作量。能不能利用已有的实验数据，从已有的二氧化硫溶解度数据估计出未知的二氧化硫溶解度数据呢？

由此引出插值问题。插值通过函数在有限个点处的取值估算出该函数在其它点处的值。设已知区间 $[a, b]$ 上的实值函数 f 在 $n+1$ 个相异点 $x_i \in [a, b]$ 处的函数值 $f_i = f(x_i)$，$i=0, 1, \cdots, n$，要求估算出 f 在 (a, b) 中某点 x 的值。插值就是用一个便于计算的简单函数 ψ 去代替 f，使得：

$$\psi(x_i) = f_i,\ i=0, 1, \cdots, n$$

并以 $\psi(x)$ 作为 $f(x)$ 近似值。称 f 为被插值函数，$x_0, x_1, \cdots, x_n$ 称为插值节点，ψ 为插值函数，$\psi(x_i)=f_i$，$i=0, 1, \cdots, n$ 是插值条件。

5.1.1 拉格朗日插值

最简单的插值莫过于依据插值条件，获得一个 n 次多项式作为插值函数 ψ，然

后利用这个 n 次多项式去估计函数 f 在其它点的值。这种插值方法被称为拉格朗日插值（Lagrange）。

MATLAB 使用 polyfit 和 polyval 两个函数来实现多项式插值。polyfit 的作用是产生插值多项式，使用格式如下：

p=polyfit(x,y,n)

x、y 是向量，分别存储插值节点与相应插值节点的函数值，n 是希望得到的多项式的次数，一般来说，对于多项式插值，n=length(x)-1。p 是插值多项式的系数向量，该向量的分量依次是该 n 次多项式从高次项到低次项的系数，直到常数项。代表获得的插值多项式。

polyval 的作用是计算多项式在指定点的函数值，基本使用格式如下：

y=polyval(p,x)

p 是代表多项式的系数向量，x 是自变量，可以是标量或向量，y 是根据 x 求出的多项式的值，如果 x 是标量，则返回一个函数值，如果 x 是向量，则 y 也是向量，y 的第 k 个分量是 x 的第 k 个分量的多项式函数的值。

下面我们看一个拉格朗日插值的例子。见表 5-2 的数据。

表 **5-2**　离散函数数据

自变量	1	2	3	4	5	6
函数值	16	18	21	17	15	12

试利用拉格朗日插值做出闭区间［1，6］的插值函数的函数图形。

MATLAB 代码如下：

```
x=1:6;                              %插值节点横坐标
y=[16 18 21 17 15 12];              %插值节点纵坐标
p=polyfit(x,y,length(x)-1);         %给出插值多项式的各项系数
tx=1:0.1:6;
ty=polyval(p,tx);                   %求插值多项式各数据点的函数值
plot(x,y,'o',tx,ty)                 %作插值多项式的函数图形
```

图形见图 5-1

从图 5-1 可以看出，拉格朗日插值获得的函数在插值区间的中间较好地反映了数据的变化趋势，但是在插值区间的两端，则出现了不合理的波动，不能反映数据的变化趋势。实验得到的数据必然存在某些误差，用多项式函数代替被插值函数也会存在误差，当多项式插值函数次数较高时，这些误差在整个插值区间将被放大和扩散，造成插值函数在插值区间内出现强烈波动，完全不能反映数据变化趋势，因此，当数据点较多时，采用拉格朗日插值估计插值区间内各点的函数值是不合适的。

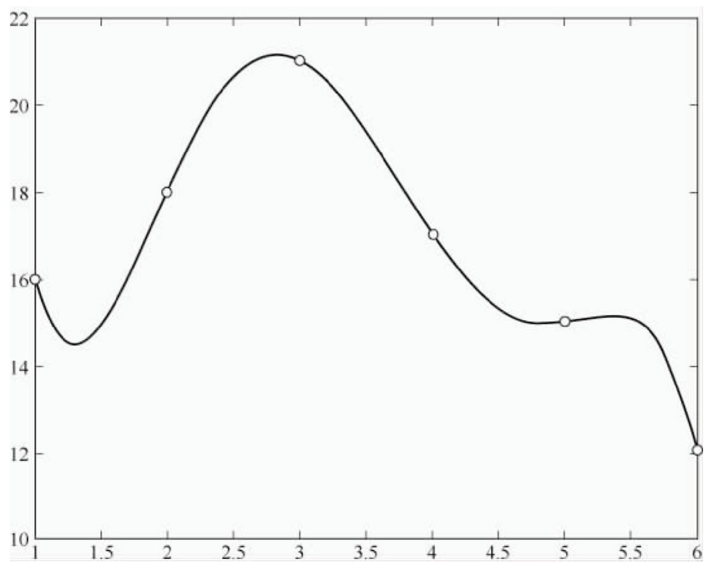


图 5-1　拉格朗日插值

由于拉格朗日插值存在上述弊端，因此提出了分段低次插值，即将插值区间剖分成若干子区间，在每个子区间采用低次多项式函数插值，这若干个低次多项式函数构成了一个插值函数，该插值函数实际上是一由低次多项式函数构成的分段函数。在每个子区间的连接处，要求插值函数连续或具有一定的光滑性。

5.1.2　分段低次插值

分段低次插值一般包括分段线性插值、分段三次厄米特插值、样条插值。

分段线性插值是在相邻插值节点之间用一线段去近似被插值函数，从整个插值区间看，则是用一条折线来做插值函数。

保形厄米特插值是在相邻插值节点之间用三次多项式近似被插值函数，整个插值函数由若干三次多项式函数构成，该插值函数在各插值节点的一阶导数由一定规则计算出来。该插值函数的特征在于能够完全反映数据点的变化趋势。

样条插值也是在相邻插值节点之间用三次多项式近似被插值函数，整个插值函数由若干三次多项式函数构成，该插值函数在各插值节点的二阶导数连续。样条插值函数的特征在于非常光滑，但反映数据变化趋势的能力不如保形厄米特插值，在插值区间的两端仍然有微小波动。

实现这三种低次分段插值的 MATLAB 函数如下：

```
yi=interp1(x,y,xi,method)
```

x、y 都是向量，分别代表插值节点的横坐标和纵坐标。x_i是自变量，可以是标量或向量，y_i是根据 x_i求出的多项式的值，如果 x_i是标量，则返回一个函数值，如果 x_i是向量，则 y_i也是向量，y_i的第 k 个分量是 x_i的第 k 个分量的多项式函数的值。

method 是一字符串，method 取不同的值，代表不同的插值方法。method 取值如下：

'linear'　　　　　线性插值（默认值）

'spline'　　　　　样条插值

'pchip'　　　　　保形厄米特插值

我们根据表 5-2 的数据，做出拉格朗日插值函数、线性插值函数、样条插值函数和保形厄米特插值函数的图形，进行比较，MATLAB 代码如下：

```
x=1:6;
y=[16 18 21 17 15 12];
tx=1:0.01:6;
subplot(2,2,1)
ty=interp1(x,y,tx,'linear');          %线性插值
plot(x,y,'o',tx,ty)
title('分段线性插值')
subplot(2,2,2)
p=polyfit(x,y,length(x)-1);           %给出拉格朗日插值多项式的各项系数
ty=polyval(p,tx);
plot(x,y,'o',tx,ty)
title('拉格朗日插值')
subplot(2,2,3)
ty=interp1(x,y,tx,'spline');          %样条插值
plot(x,y,'o',tx,ty)
title('样条插值')
subplot(2,2,4)
ty=interp1(x,y,tx,'pchip');           %保形厄米特插值
plot(x,y,'o',tx,ty)
title('保形厄米特插值')
```

上述 MATLAB 代码用到了 subplot 命令，该命令的使用格式如下：

```
subplot (m,n,p)
```

它的作用是将整个图形窗口分割为 m 行×n 列的小图形窗口，在第 p 个子图形窗口作图。做出的图形见图 5-2，可以看出，拉格朗日插值图形在插值区间两端出现了不合理的波动，线性插值虽然在整个插值区间反映了数据的变化趋势，但没有光滑性。保形厄米特插值也在整个插值区间反映了数据的变化趋势，具有一定的光滑度，但保形厄米特插值依据一定规则计算出各插值节点的一阶导数值，这不一定是合理的。样条插值具有最高的光滑度，较好地反映了数据变化趋势，但在插值区

间两端具有微小波动，样条插值认为各插值节点的二阶导数连续，这样的假定具有一定合理性。

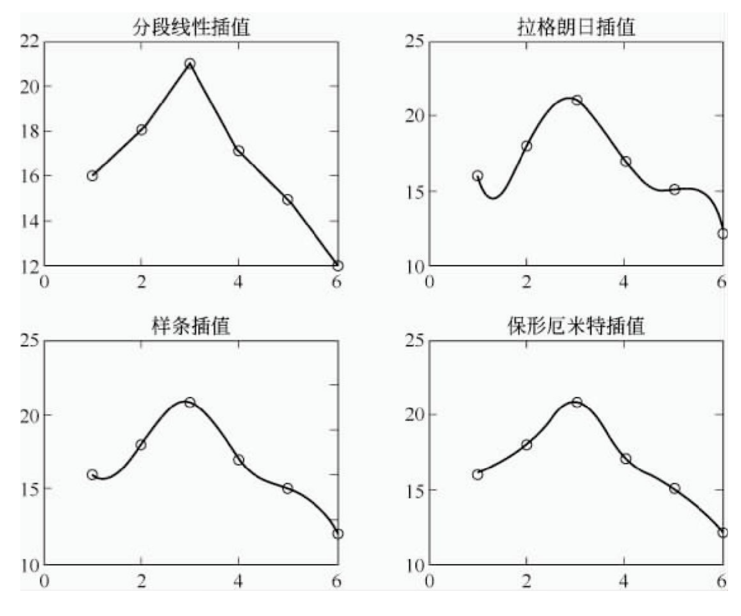


图 5-2　常用插值方法的比较

5.1.3　山丘基样条插值

保形厄米特插值能够反映数据的变化趋势，但不如样条插值得到的曲线光滑，并且依据一定规则计算出各插值节点的一阶导数值不一定是合理的。样条插值在插值区间两端有微小波动，但非常光滑，而且样条插值认为各插值节点的二阶导数连续，这样的假定具有一定合理性。能否将这两种插值方法结合起来，获得既能反映数据变化趋势，又充分光滑的插值函数呢？答案是肯定的。

具有 $n+1$ 个插值节点的样条插值函数可以由所谓山丘基样条函数 $\psi_i(x)$ 组合而成：

$$f(x) = \sum_{i=-3}^{n-1} a_i \psi_i(x)$$

$$\psi_i(x) = 0.5 \sum_{j=i}^{i+4} \beta_{i,j} \left| x - t_j \right|^3$$

a_i 是系数，共有 $n+3$ 个，t_j 是插值节点，$\beta_{i,j}$ 由插值节点确定。因此，只要能够确定系数 a_i，问题也就迎刃而解了。由 $n+1$ 个插值节点的函数值，可以确定 $n+1$ 个线性方程，但未知数有 $n+3$ 个，所以还需要确定两个方程，这剩余的两个方程我们通过指定插值区间的两个端点的一阶导数值来给出。我们使用保形厄米特插值方法中确定插值区间两个端点处的一阶导数值的方法来确定本问题插值区间端点的

一阶导数值。这样，方程的数目和未知数的数目相等了，我们可以求解方程，得到 a_i，从而解决了山丘基样条的插值问题。由于山丘基样条插值采用了山丘基样条函数的组合，因此该插值函数仍然保留了样条函数的特性，在各插值节点处二阶导数连续，具有很高的光滑性。同时确定插值区间两个端点处的一阶导数值采用了保形厄米特插值中确定插值区间端点处一阶导数值的方法，克服了插值区间两端处的不合理波动。

完成山丘基样条插值的 MATLAB 代码如下：

```
function y0=hillspline3(x0,x,y)
%    已知离散点值,利用 3 次紧凑山丘基样条插值获得函数值 y0
%    x0:待估计点的横坐标
%    x,y:插值节点,是同类型,相同长度的向量
if size(x,1)~=1
   x=x';
end
if size(y,1)~=1
   y=y';
end
[a,t]=hillsplinex(x,y); a=a'; t=t';
for i=1:length(t)-4
     phi(i)=powerplus3(x0,t(i:i+4));
end
y0=sum(a. * phi);
function [ax,tx]=hillsplinex(t,y)
%t  已知点横坐标,行向量,长度 n
%y  已知点纵坐标,行向量
%ax  基函数的系数  列向量 n+2 个
%tx  扩展点横坐标  列向量 n+6 个
            h=diff(t); delta=diff(y)./h; n=length(h)+1;
            d1=pchipendpoint(h(1),h(2),delta(1),delta(2));
            dn=pchipendpoint(h(n-1),h(n-2),delta(n-1),delta(n-2));
            r=[d1;y';dn];
            tx=[t(1)-[1:3] * h(1),t,t(length(t))+[1:3] * h(length(h))];
            a=zeros(length(tx)-4,length(tx)-4);
            for i=1:3
```

```
            a(1,i)=depowerplus3(t(1),tx(i:i+4));
        end
        for i=length(tx)-6:length(tx)-4
            a(length(tx)-4,i)=depowerplus3(t(length(t)),tx(i:i+4));
        end
        for j=2:length(tx)-5
            for i=j-1:j+1
                a(j,i)=powerplus3(t(j-1),tx(i:i+4));
            end
        end
        ax=a\r; tx=tx';
function y=powerplus3(x,t)
%x 点横坐标
%t 点横坐标 长度为5的横向量
%y 基函数值
        c=t;
        for i=1:5
            c(i)=[];
            beta(i)=24/prod(t(i)-c);
            c=t;
        end
        powerplus=abs(x-t).^3;
        y=0.5 *sum(beta.*powerplus);
function y=depowerplus3(x,t)
%x 点横坐标
%t 点横坐标 长度为5的横向量
%y 基函数的导数值
        c=t;
        for i=1:5
            c(i)=[];
            beta(i)=24/prod(t(i)-c);
            c=t;
        end
        powerplus=3 *sign(x-t).*(x-t).^2;
        y=0.5 *sum(beta.*powerplus);
```

```
function d=pchipendpoint(h1,h2,del1,del2)
% 确定区间端点导数值
% 区间端点处两子区间的长度
% 区间端点处两子区间处的折线的斜率
d=((2 *h1+h2)*del1-h1 *del2)/(h1+h2);
if sign(d)~=sign(del1)
            d=0;
elseif(sign(del1)~=sign(del2))&(abs(d)>abs(3 *del1))
            d=3 *del1;
end
```

完成山丘基样条插值的主函数自然是 hillspline3，在主函数 hillspline3 中，调用了若干子函数，这些子函数的代码及简要说明均列出。Hillsplinex 函数中用到了 diff 函数，diff 函数是 MATLAB 的固有函数，当 X 是向量时，diff(x)返回:[$X(2)-X(1)\quad X(3)-X(2)\cdots X(n)-X(n-1)$]。

为了完成山丘基样条插值，需要把上述函数分别单独保存在 MATLAB 的安装目录下的 work 子目录中，然后就可以运行主函数 hillspline3 进行山丘基样条插值了。

我们仍然引用上面的例子，看一看山丘基样条插值的结果。具体 MATLAB 代码：

```
x=1:6;
y=[16 18 21 17 15 12];
tx=1:0.01:6;
ty=zeros(1,length(tx));
for k=1:length(tx)
    ty(k)=hillspline3(tx(k),x,y);
end
plot(x,y,'o',tx,ty)
```

做出的山丘基样条插值函数图形见图 5-3。

从图 5-3 可以看出，利用山丘基样条插值获得插值函数，既有很高的光滑性，又反映了数据的变化趋势，克服了原先样条插值在插值区间两端处的微小波动问题。

5.1.3.1 估计二氧化硫的溶解度数据

对于本章开头举的例子，利用山丘基样条插值估计当液面上 SO_2 分压为 8.4mmHg 时，二氧化硫的溶解度数据。MATLAB 代码如下：

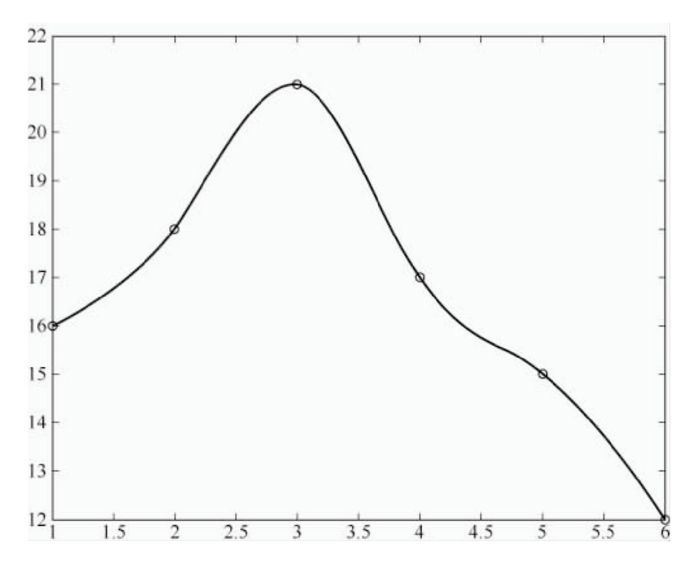


图 5-3　山丘基样条插值函数图形

```
x=[5.8 14.7 26 39 59];
y=[0.15 0.3 0.5 0.7 1];
x0=8.4;                       %液面上SO2分压/mmHg
y0=hillspline3(x0,x,y)        %对应的二氧化硫的溶解度数据/kg SO2/100kg H2O
```

结果如下：

y0=0.1930

实验测定值是 0.2kg SO_2/100kg H_2O，可见山丘基样条插值估计精度很高，相对误差 3.6%，小于 5%。

5.1.3.2　估计水的黏度

黏度是流体的重要物理性质之一，今有水的黏度数据见表 5-3，要求根据表中的数据估计出水在 50℃时的黏度。

表 **5-3**　水的黏度数据

温度/℃	0	10	20	30	40	60	70	80	90	100
黏度/mPa·s	1.789	1.305	1.005	0.801	0.653	0.470	0.406	0.355	0.315	0.283

我们采用山丘基样条插值估计水在 50℃时的黏度，代码如下：

```
t=[0:10:40 60:10:100];                                                    %温度数据
viscosity=[1.789 1.305 1.005 0.801 0.653 0.470 0.406 0.355 0.315 0.283];  %黏度数据
t_50=50;
viscosity_50=hillspline3(t_50,t,viscosity)
```

结果是：

viscosity _50＝0.5479

水在50℃时的黏度的测量值是0.549mPa·s，估计值和测量值的相对误差仅为0.21％。

很多物性数据都是通过列表形式给出，是一些离散数据，可以利用山丘基样条插值估计列表中没有给出物性数据，并且能得到很高的准确度。

5.2　数值微分与数值积分

微积分是化学工作者认识化学规律的强有力工具，在化学领域具有广泛应用。但微积分运算属于超越运算，本质上是一个求极限的过程，与普通的算术运算具有很大不同，因此手动进行微积分运算，难度比较高、容易出错。在具体科研与教学中遇到的微积分问题包括两类，一类是对具有明确解析式的函数进行积分或微分，另一类是对仅由一些离散数据点表达的函数进行微分或积分。在本章，我们对这两类函数的微分和积分均进行讨论，特别是对后一类函数进行微分或积分，在化学数据处理中具有特别重要的意义。

5.2.1　具有明确解析式的函数的微分与积分

我们知道，微分实际上是函数增量与自变量增量之比的极限，因此，最简单的求取微分的方法就是用增量比的近似值，即差商作为微分的近似值，如求函数$f(x)$在x_0点的微分，可以用差商近似：

$$f'(x_0)\approx\frac{f(x_0+h)-f(x_0)}{h}$$

h是很小的正数。采用差商方法，当h比较小时，能够求得函数$f(x)$在x_0点的微分的近似值。表面看来，根据微分的定义，h越小，这种方法求得的微分近似值的精度越高，因此h越小越好。但在实际计算中，问题远不是那么简单，由于计算机本身存储数据的位数是有限的，因此实际计算出来的$f(x_0+h)-f(x_0)$的值与真实值是不同的，具有一定的误差e，当h过小时，请注意h位于分母，误差e被急剧放大，从而使得整个微分计算结果产生较大误差。因此，差商方法求微分对h的要求是矛盾的，因此差商方法不是求微分的好方法。

求微分通常采用外推方法。

我们知道：

$$f'(x_0)=\lim_{h\to 0}\frac{f(x_0+h)-f(x_0)}{h}$$

我们令：

$$F(h)=\frac{f(x_0+h)-f(x_0)}{h}$$

如果我们令 $h=1$，1/2，1/4，…，1/64，一共可以求得 7 个 $F(h)$的函数值，这样我们就有了 7 个插值节点，我们根据这 7 个插值节点，可以构造一个关于 h 的 6 次插值多项式 $p(h)$，此时插值多项式 $p(h)$在 $h=0$ 处的函数值 $p(0)$，就是函数 $f(x)$在 x_0 点的微分值。构造插值多项式 $p(h)$的那些插值节点，被称为外推序列，采用不同的外推序列能够得到不同精确度的微分值。使用外推方法必须构造插值多项式，前一章讲述的构造拉格朗日插值多项式的方法当然是可以使用的，但对于外推方法而言，构造外推多项式 $p(h)$的更好方法是采用 Neville 方法，对于 Neville 方法，我们这里不讲，我们仅讲述外推方法的基本原理，如需要了解 Neville 方法的详细内容，请读者参考有关数学教材。

利用外推方法求微分的 MATLAB 函数如下：

```
function y=diff_m(f_name,x,tolerance)
%外推方法求微分
        m=nargin;
        if m<3
           tolerance=1e-6;
        end
        m=0;
        h1=1;
        d1=(feval(f_name,x+h1)-feval(f_name,x-h1))/(2 *h1);
        while 1
            [h2,d2]=recurdiff(f_name,x,h1,d1);
            L1=abs(d2(length(d2))-d1(length(d1)))<tolerance;
            L2=2 *abs(d2(length(d2))-d1(length(d1)))/(abs(d2(length
(d2)))+abs(d1(length(d1))))<tolerance;
            if L1&L2                       %外推终止条件
                y=d2(length(d2));
                return
            else
                h1=h2;d1=d2;
            end
            m=m+1;
            if k==100
               y=inf;return
            end
        end
```

```
function [h,d]=recurdiff(f_name,x,h1,d1)
                h=h1/2;
                d(1)=(feval(f_name,x+h)-feval(f_name,x-h))/(2 * h);
                for k=2:length(d1)+1
                    d(k)=(4^(k-1) * d(k-1)-d1(k-1))/(4^(k-1)-1);
                end
```

我们举例说明利用 diff _m 函数求微分，需要注意的是 diff _m 函数调用了 recurdiff 函数。

求函数 $y=e^x$ 在 $x=1$ 点的导数值。实现定义函数 $y=e^x$：

```
function y=f(x)
        y=exp(x);
```

然后使用 diff _m 函数：

```
format long
y=diff_m('f',1)
```

结果是：

y=2.71828182845911

数据结果用 long 格式显示，以和真实值进行对比。函数 $y=e^x$ 在 $x=1$ 点的导数值的真实值是 2.71828182845905，可见利用外推方法求函数微分的精确度很高。

多元函数的偏导数及函数向量的雅科比矩阵在科学计算中也有广泛的用途，我们在外推方法求微分的 diff _m 函数的基础上，编写求多元函数的偏导数的 MATLAB 函数及求函数向量的雅科比矩阵的 MATLAB 函数。

```
function y=diff_xy(f_name,x0,k,tolerance)
          % 求多元函数 f(x)在 x0 的,对第 k 个变量的偏导数
          % f_name:M 文件定义的函数
          % k:对第 k 个变量的偏导数
          % tolerance:误差限,可选择参数,默认值 1e-6
          m=nargin;
          if m<4
             tolerance=1e-6;
          end
          m=0;
          h1=1;
```

```
        x2=x0;x2(k)=x0(k)+h1;
        x1=x0;x1(k)=x0(k)-h1;
        d1=(feval(f_name,x2)-feval(f_name,x1))/(2 * h1);
        while 1
            [h2,d2]=recurdiff_xy(f_name,x0,k,h1,d1);
            L1=abs(d2(length(d2))-d1(length(d1)))<tolerance;
            L2=2 * abs(d2(length(d2))-d1(length(d1)))/(abs(d2(length
(d2)))+abs(d1(length(d1))))<tolerance;
            if L1&L2          %外推终止条件
                y=d2(length(d2));
                return
            else
                h1=h2;d1=d2;
            end
            m=m+1;
            if m==100
               y=inf;return
            end
        end
    function [h,d]=recurdiff_xy(f_name,x0,k,h1,d1)
            h=h1/2;
            x2=x0;x2(k)=x0(k)+h;
            x1=x0;x1(k)=x0(k)-h;
            d(1)=(feval(f_name,x2)-feval(f_name,x1))/(2 * h);
            for k=2:length(d1)+1
                d(k)=(4^(k-1) * d(k-1)-d1(k-1))/(4^(k-1)-1);
            end
```

diff_xy 是求多元函数的偏导数的主函数，它调用了 recurdiff_xy 子函数。

```
    function y=matrix_jocbi(f_name,x0,tolerance)
        %求函数向量在 x0 的雅可比矩阵
        %f_name 为函数向量,由一个 M 文件定义
        % tolerance:误差限,可选择参数,默认值 1e-6
        m=nargin;
        if m<3
```

```
        tolerance=1e-6;
    end
    L=length(feval(f_name,x0));
    k=length(x0);
    for k1=1:L
        h=@(x) pick(feval(f_name,x),k1);
        for k2=1:k
            y(k1,k2)=diff_xy(h,x0,k2,tolerance);
        end
    end
function y=pick(a,k)
    y=a(k);
```

函数向量由 M 文件定义，定义格式是［f1 f2…fn］或［f1;f2;…;fn］。求雅科比矩阵的主函数是 matrix_jocbi，它调用了子函数 pick。子函数 pick 的作用是返回常数向量的某个分量。在 matrix_jocbi 函数中，使用了语句 h=@(x)pick(feval(f_name,x),k1)，该语句的作用是用 h 代表以 x 为自变量的函数 pick(feval(f_name,x),k1)，h 被称为该函数的句柄，feval(h,x)的作用相当于 pick(feval(f_name,x),k1)。

下面我们求函数向量[$e^{x+y}+\cos(xy)x\sin y$]在点［1.1 −3.6］的雅科比矩阵。

首先定义函数向量：

```
function y=f(x)
    y(1)=exp(sum(x))+sin(prod(x));
    y(2)=x(1)*sin(x(2));
```

上述定义函数向量的函数中用到了 prod 函数，但 x 是向量时，prod(x)函数返回向量 x 各分量的积。

在完成函数向量的定义后，我们使用 matrix_jocbi 函数求该函数向量的雅科比矩阵：

matrix_jocbi('f',[1.1 −3.6])

运行结果是

```
    2.5423  -0.6696
ans=
    0.4425  -0.9864
```

对于在［a，b］的实函数 $f(x)$，利用计算机求其定积分时，一般是将 $f(x)$作为［a，b］的被插值函数，找到 $f(x)$的插值函数 $F(x)$，用 $F(x)$在［a，b］的积分值作为 $f(x)$在［a，b］的积分值：

$$\int_a^b f(x) \approx \int_a^b \Phi(x)$$

比如在［a，b］上 $f(x)$的插值函数 $F(x)$可以是闭区间［a，b］上一条插值折线。以定积分 $I=\int_0^1 x^2 e^x dx$ 的计算为例，我们说明积分计算原理，积分计算原理见图 5-4。

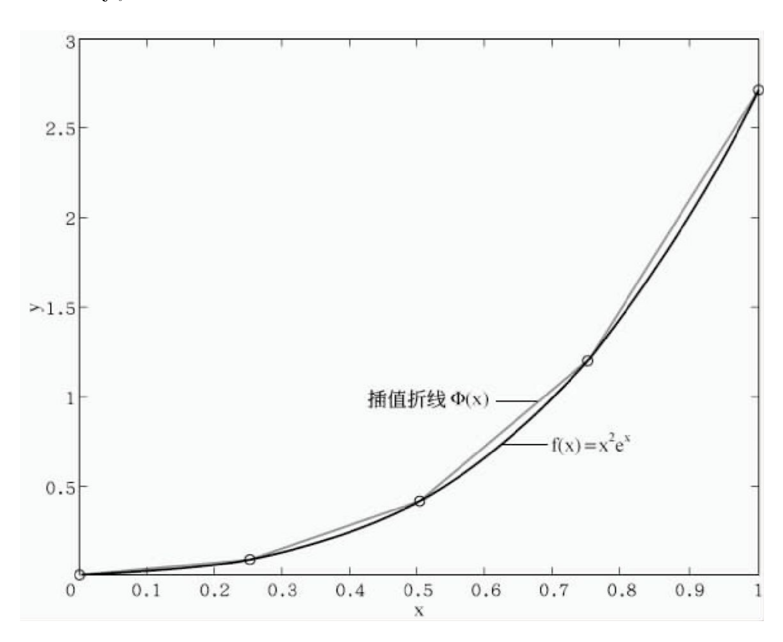


图 5-4　积分计算原理

图 5-4 采用 MATLAB 进行作图，代码如下：

```
x=0:0.1:1;
y=x.^2.*exp(x);
tx=0:0.25:1;
ty=tx.^2.*exp(tx);
plot(x,y,tx,ty,'-o')
xlabel('x')
ylabel('y')
text(0.42,1,'插值折线 F(x)\rightarrow')
                              %在[0.42,1]处添加文本"插值折线 Φ(x)→"
text(0.64,0.75,'\leftarrow f(x)=x^2e^x')
                              %在[0.64,0.75]处添加文本"←y=x²eˣ"
```

需要注意的是 text 函数，text 函数的作用是在做出的图形上写上指定的文本内容。text(0.64,0.75,'\leftarrow y=x^2e^x')的作用是在图形的指定位置处（坐标

[0.64，0.75]）写入文本“←y=x²eˣ”，\leftarrow 代表左箭头“←”，跟在^后面的第一个字符采用上标形式，如 x^2 代表“x²”。

$y=x^2e^x$ 是不容易积分的，但在 [0，1] 的它的插值折线由线段分段构成，直线的积分非常容易，只要 [0，1] 的插值节点足够多，由插值折线计算出来的积分值将会越来越接近 $y=x^2e^x$ 在 [0，1] 的积分值，最终解决有明确解析式的函数的积分问题。

利用插值折线的积分值去近似原来的函数积分值的方法，被称为梯形积分方法。

如果将积分区间偶数等分，获得奇数个插值节点，从积分区间的左端点开始，每三个插值节点确定一条二次抛物线，结果插值函数将是一分段的二次抛物线，然后对该分段的二次抛物线进行积分，获得积分值。这被称为辛普森方法。

以上简要说明计算机求解积分的原理，MATLAB 采用 quadl 函数计算定积分。quadl 函数的使用格式是：

quadl(f,a,b)

f 是字符串，代表函数名。a，b 分别是积分区间的上限和下限。特别需要注意的是 quadl 函数要求定义被积函数时，函数文件中的各种运算符采用点运算符。

我们以 $I=\int_0^1 x^2e^x dx$ 为例，用 quadl 函数求解上述积分。

首先定义函数：

```
function y=f(x)
        y=x. *x . *exp(x);
```

然后用 quadl 函数求解，代码如下：

```
format long
I=quadl('f',0,1)
```

结果如下：

I=0.71828182847289

积分的精确值是 0.718281828，可见 quadl 函数积分精度很高。

5.2.2　由离散数据点表达的函数的微分或积分

我们在实验中取得数据通常是一些离散的数据点，这些数据点同样反映了一种函数关系。如在化学动力学实验中测得的不同时刻反应物的浓度数据。如果想要求得任一时刻的反应速率，就需要浓度对时间进行微分（封闭体系）。这就引出了由离散数据点表达的函数的微分的问题。

对于无相变和化学变化且不做非膨胀功的均相封闭系统，它的焓变可以表达为：

$$\Delta H=\int C_p dT$$

等压热容随温度而变化，为了求得系统的焓变，必须用等压热容对温度积分。如果我们只有一些不同温度下该系统的热容数据而没有明确的等压热容随温度变化的解析表达式，该如何积分呢？这就是由离散数据点表达的函数的积分的问题。

如果我们能够由把离散数据点表达的函数转换为具有明确解析式的函数，那么，进行微分或积分就方便了。由此我们想到把用离散数据点表达的函数通过山丘基样条插值转变为具有明确解析式的函数，然后进行微分或积分。由插值节点获得的山丘基样条函数二阶导数连续，是三次多项式的组合，能够很方便地进行微分或积分。

$$f(x) = \sum_{i=-3}^{n-1} a_i \psi_i(x)$$

$$\psi_i(x) = 0.5 \sum_{j=i}^{i+4} \beta_{i,j} \left| x - t_j \right|^3$$

从山丘基样条插值函数的表达方式上看，对其微分或积分，实际就是对函数 $\left| x-t_j \right|^3$ 进行微分或积分。

微分结果是：$3 * \mathrm{sign}(x-x_j) \left| x-t_j \right|^2$

积分结果是：$0.25 * \mathrm{sign}(x-x_j) \left| x-t_j \right|^4$（假设积分常数为0）

sign是符号函数，当 $x-t_j>0$ 时返回1，当 $x-t_j<0$ 时返回−1。

求离散函数积分的主函数是inhillspline3，它调用其它函数，共同解决这个问题。

```
function I=inhillspline3(x1,x2,x,y)
%    已知离散点值,利用3次紧凑山丘基样条插值获得函数定积分值I
%    x1,x2:积分区间的下限和上限
%    x,y是同类型,相同长度的向量,插值节点
if size(x,1)~=1
    x=x';
end
if size(y,1)~=1
    y=y';
end
%a  基函数的系数 行向量
%t  扩展点横坐标 行向量
[a,t]=hillsplinex(x,y);
a=a';t=t';
for i=1:length(t)-4
      phi(i)=inpowerplus3(x1,t(i:i+4));
end
```

```
I1=sum(a.*phi);
for i=1:length(t)-4
    phi(i)=inpowerplus3(x2,t(i:i+4));
end
I2=sum(a.*phi);
I=I2-I1;
function [ax,tx]=hillsplinex(t,y)
%t  已知点横坐标，行向量，长度 n
%y  已知点纵坐标，行向量
%ax  基函数的系数  列向量 n+2 个
%tx  扩展点横坐标  列向量 n+6 个
         h=diff(t);
         delta=diff(y)./h;
         n=length(h)+1;
         d1=pchipendpoint(h(1),h(2),delta(1),delta(2));
         dn=pchipendpoint(h(n-1),h(n-2),delta(n-1),delta(n-2));
         r=[d1;y';dn];
         tx=[t(1)-[1:3]*h(1),t,t(length(t))+[1:3]*h(length(h))];
         a=zeros(length(tx)-4,length(tx)-4);
         for i=1:3
             a(1,i)=depowerplus3(t(1),tx(i:i+4));
         end
         for i=length(tx)-6:length(tx)-4
             a(length(tx)-4,i)=depowerplus3(t(length(t)),tx(i:i+4));
         end
         for j=2:length(tx)-5
             for i=j-1:j+1
                 a(j,i)=powerplus3(t(j-1),tx(i:i+4));
             end
         end
         ax=a\r;
         tx=tx';
function y=powerplus3(x,t)
%x  点横坐标
%t  点横坐标  长度为 5 的横向量
```

```
%y 基函数值
        c=t;
        for i=1:5
            c(i)=[];
            beta(i)=24/prod(t(i)-c);
            c=t;
        end
        powerplus=abs(x-t).^3;
        y=0.5 * sum(beta. * powerplus);
function y=depowerplus3(x,t)
%x 点横坐标
%t 点横坐标 长度为5的横向量
%y 基函数的导数值
        c=t;
        for i=1:5
            c(i)=[];
            beta(i)=24/prod(t(i)-c);
            c=t;
        end
        powerplus=3 * sign(x-t). * (x-t).^2;
        y=0.5 * sum(beta. * powerplus);
function y=inpowerplus3(x,t)
%x 点横坐标
%t 点横坐标 长度为5的横向量
%y 基函数的积分值，设积分常数为0
        c=t;
        for i=1:5
            c(i)=[];
            beta(i)=24/prod(t(i)-c);
            c=t;
        end
        powerplus=0.25 * sign(x-t). * (x-t).^4;
        y=0.5 * sum(beta. * powerplus);
function d=pchipendpoint(h1,h2,del1,del2)
% 确定区间端点导数值
```

```
% 区间端点处两子区间的长度
% 区间端点处两子区间处的折线的斜率
d=((2 *h1+h2)*del1 - h1 *del2)/(h1+h2);
if sign(d)~=sign(del1)
            d=0;
elseif (sign(del1) ~=sign(del2)) & (abs(d) >abs(3 *del1))
            d=3 *del1;
end
```

dehillspline3 函数是求由离散数据点表达的函数的微分的主函数，它调用其它函数共同解决这个问题。

```
function y0=dehillspline3(x0,x,y)
%    已知离散点值，利用 3 次紧凑山丘基样条插值获得函数导数值 y0
%    x0:未知点横坐标
%    x,y 是同类型,相同长度的向量
if size(x,1)~=1
   x=x';
end
if size(y,1)~=1
   y=y';
end
%a 基函数的系数 行向量
%t 扩展点横坐标 行向量
[a,t]=hillsplinex(x,y);
a=a';t=t';
for i=1:length(t)-4
     phi(i)=depowerplus3(x0,t(i:i+4));
end
y0=sum(a. *phi);
function [ax,tx]=hillsplinex(t,y)
%t 已知点横坐标，行向量，长度 n
%y 已知点纵坐标，行向量
%ax 基函数的系数 列向量 n+2 个
%tx 扩展点横坐标 列向量 n+6 个
          h=diff(t);
```

```
        delta=diff(y)./h;
        n=length(h)+1;
        d1=pchipendpoint(h(1),h(2),delta(1),delta(2));
        dn=pchipendpoint(h(n-1),h(n-2),delta(n-1),delta(n-2));
        r=[d1;y';dn];
        tx=[t(1)-[1:3]*h(1),t,t(length(t))+[1:3]*h(length(h))];
        a=zeros(length(tx)-4,length(tx)-4);
        for i=1:3
            a(1,i)=depowerplus3(t(1),tx(i:i+4));
        end
        for i=length(tx)-6:length(tx)-4
            a(length(tx)-4,i)=depowerplus3(t(length(t)),tx(i:i+4));
        end
        for j=2:length(tx)-5
            for i=j-1:j+1
                a(j,i)=powerplus3(t(j-1),tx(i:i+4));
            end
        end
        ax=a\r;
        tx=tx';
function y=powerplus3(x,t)
%x 点横坐标
%t 点横坐标 长度为5的横向量
%y 基函数数值
        c=t;
        for i=1:5
           c(i)=[];
           beta(i)=24/prod(t(i)-c);
           c=t;
        end
        powerplus=abs(x-t).^3;
        y=0.5*sum(beta.*powerplus);
function y=depowerplus3(x,t)
%x 点横坐标
%t 点横坐标 长度为5的横向量
```

```
%y 基函数的导数值
        c=t;
        for i=1:5
            c(i)=[];
            beta(i)=24/prod(t(i)-c);
            c=t;
        end
function d=pchipendpoint(h1,h2,del1,del2)
% 确定区间端点导数值
% 区间端点处两子区间的长度
% 区间端点处两子区间处的折线的斜率
d=((2 *h1+h2)*del1 - h1 *del2)/(h1+h2);
if sign(d) ~=sign(del1)
            d=0;
elseif (sign(del1) ~=sign(del2)) & (abs(d) >abs(3 *del1))
            d=3 *del1;
end
```

下面我们举两个例子，说明求由离散数据点表达的函数的微分在化学中的应用。

5.2.2.1　气相分解反应动力学

考虑等温气相分解反应：

$$A \longrightarrow B + C$$

见表 5-4，它给出的实验结果表明了作为转化率函数的化学反应数据。温度 422.2K，总压为 10atm，初始原料为 A 和惰性气体各占 50% 的混合物。求当转化率达到 80% 时，需要多长的反应时间。

表 5-4　作为转化率函数的化学反应数据

x	$-r_A$/[mol/(L·s)]	x	$-r_A$/[mol/(L·s)]
0	0.0053	0.5000	0.0033
0.1000	0.0052	0.6000	0.0025
0.2000	0.0050	0.7000	0.0018
0.3000	0.0045	0.8000	0.0013
0.4000	0.0040	0.8500	0.0010

根据动力学方程和$[A]=[A_0]*(1-x)$，得到：

$-d[A]/dt=[A_0]dx/dt=-r_A$

积分，得：

$$t=[A_0]\int_0^x \frac{\mathrm{d}x}{-r_A}$$

因此，求得反应时间的关键是求得上式右面的积分，由于我们只是知道 $1/(-r_A)$ 在不同转化率下的离散值，因此必须对由离散数据点点表达的函数 $1/(-r_A)$ 积分，方可求得达到 80%转化率所需要的时间。解决该问题的 MATLAB 代码如下：

```
x=[0:0.1:0.8 0.85];
rA=-[53 52 50 45 40 33 25 18 12.5 10]/10000;
P=10 *101325;
T=422.2;
R=8.314;
C=P/(R *T);
Ca0=0.5 *C;
I=inhillspline3(0,0.8,x,1./(-rA));
t=Ca0/1000 *I
```

结果是：

t=37.3696

因此，该气相反应在指定温度、压力和初始浓度条件下，在第 37.4s 达到 80%的转化率。

5.2.2.2 气相色谱数据处理

某一化学混合物用气相色谱法分离，在气相色谱图上出现了 A、B 两个明显的峰，将测出的峰形作为保留时间的函数，得到表 5-5 的数据，由这些数据求出混合物中 A、B 两物质的相对量。

表 5-5　气相色谱数据

物质 A		物质 B	
强度(任意单位)	时间/秒	强度(任意单位)	时间/秒
3	42	0.5	250
17	53	6	258
40	64	14	266
69	75	21	274
90	86	13	282
68	97	5	290
43	108	0	298
19	119		
4	130		

A、B 的相对量之比即为它们色谱峰下的面积之比，强度对时间进行积分就获得色谱峰的面积，在本问题中，强度不是对时间的连续曲线，而是离散的数据点，因此，必须采用对由离散数据点表达的函数进行积分。具体的 MATLAB 代码如下：

```
xA=[3 17 40 69 90 68 43 19 4];
tA=[42 53 64 75 86 97 108 119 130];
xB=[0.5 6 14 21 13 5 0];
tB=[250 258 266 274 282 290 298];
A=inhillspline3(42,130,tA,xA);
B=inhillspline3(250,298,tB,xB);
A_percent=A/(A+B)
B_percent=B/(A+B)
```

结果是：

A _percent=0.8896

B _percent=0.1104

5.2.2.3　三甲胺和溴丙烷的反应动力学

在一间歇反应器里研究三甲胺和溴丙烷的反应动力学：

$$\underset{A}{N(CH_3)_3}+\underset{B}{CH_3CH_2CH_2Br}\longrightarrow\underset{C}{(CH_3)_3(CH_2CH_2CH_2)NBr}$$

当反应温度为 139.4℃时，三甲胺和溴丙烷初始浓度均为 0.1mol/L，不同反应时间的转化率见表 5-6，求反应的总级数和反应速率常数。

表 5-6　三甲胺不同反应时间的转化率

t/min	0	13	34	59	120
x/%	0	11.2	25.7	36.7	55.2

由于三甲胺和溴丙烷初始浓度相同，并且它们的化学计量系数相同，因此任一时刻，它们的浓度相等，因此反应动力学方程可以写为：$-dC_A/dt=-ra=k_AC_A^{\alpha+\beta}$

动力学方程两边取自然对数：$Ln(-dC_A/dt)=Lnk_A+(\alpha+\beta)LnC_A$

反应总级数 $n=\alpha+\beta$。

如果能求出三甲胺的浓度对上表中所给任一时刻的导数值，我们就得到一个由 5 个方程构成的线性矛盾方程组（未知量为 Lnk_A和 n）。从而可求得 k_A和 n。

具体 MATLAB 代码如下：

```
t=[0 13 34 59 120]*60;              %记录数据的时刻，单位为秒
x=[0 11.2 25.7 36.7 55.2]/100;      %相应时刻的三甲胺转化率
CA0=0.1;
CA=CA0 *(1-x);                      %三甲胺在相应时刻的浓度
ra=zeros(1,length(x));
for k=1:length(x)
```

```
ra(k)=dehillspline3(t(k),t,CA);         %三甲胺浓度对相应时刻的导数值
 end
 b=log(-ra');
 a=[ones(length(x),1),log(CA')];        %构造线性矛盾方程组的系数矩阵
 z=a\b;                                 %求解线性矛盾方程组
 kA=exp(z(1))                           %反应速率常数
 n=z(2)                                 %反应总级数
```

结果是：

kA=0.0014

n=1.9366

从反应总级数上，n 接近 2，因此反应总级数为 2。

我们对反应总级数做了修正，因此要根据修正的反应总级数再次计算速率常数 k，对于 2 级反应：

$$k_A t = 1/C_A - 1/C_{A0}$$

我们有 5 组不同时刻三甲胺的浓度数据，因此可以得到一个矛盾线性方程组（未知量为 k_A），求得速率常数 k_A。

```
t=[0 13 34 59 120]*60;
x=[0 11.2 25.7 36.7 55.2]/100;
CA0=0.1;
CA=CA0 *(1-x);
y=[1./CA-1/CA0]';                   %构造线性矛盾方程组的系数矩阵
kA=t'\y                             %求解线性矛盾方程组
```

结果为：

k_A=0.0017

所以反应的总级数是 2，速率常数 k_A 为 0.0017L/(mol·s)。

5.2.2.4　电位滴定终点的确定

当滴定反应平衡常数较小，滴定突跃不明显，或试液有色、混浊，用指示剂指示终点有困难时，可以采用电位滴定方法，即根据滴定过程中等当点附近的电位突跃来确定终点。只要选择合适的电极，电位滴定可以用于酸碱滴定、沉淀滴定、氧化还原滴定和络合滴定。

在滴定过程中，每加一次滴定剂，测量一次电动势，直到超过等当点为止。这样就得到一系列的滴定剂用量（v）和相应电动势（E）的数值。利用一系列 E-v 数据，获得 dE/dv-v 曲线，该曲线的最高点对应的 v 值，就是滴定终点。因此，确定电位滴定的关键之一就是能够求得电位 E 对加入的滴定剂体积的导数值。

我们以 0.1mol/L 的 $AgNO_3$ 溶液滴定 NaCl 溶液的滴定数据为例，确定该电位滴定的滴定终点。指示电极是银电极，参比电极是饱和的甘汞电极。得到的 0.1mol/L 的 $AgNO_3$ 溶液滴定 NaCl 溶液的电位滴定 E-v 数据见表 5-7。

表 5-7　0.1mol/L 的 $AgNO_3$ 溶液滴定 NaCl 溶液的电位滴定数据

v(AgNO_3)/mL	电动势 E/v	v(AgNO_3)/mL	电动势 E/v
5.0000	0.0620	24.2000	0.1940
15.0000	0.0850	24.3000	0.2330
20.0000	0.1070	24.4000	0.3160
22.0000	0.1230	24.5000	0.3400
23.0000	0.1380	24.6000	0.3510
23.5000	0.1460	24.7000	0.3580
23.8000	0.1610	25.0000	0.3730
24.0000	0.1740	25.5000	0.3850
24.1000	0.1830		

我们根据 E-v 数据首先做出 dE/dv-v 曲线，自然需要用到 dehillspline3 求数值微分，具体的 MATLAB 代码如下：

```
v=[5 15 20 22 23 23.5 23.8 24 24.1 24.2 24.3 24.4 24.5 24.6 24.7 25 25.5];
E1=[0.062 0.085 0.107 0.123 0.138 0.146];
E2=[0.161 0.174 0.183 0.194 0.233 0.316 0.340 0.351 0.358 0.373 0.385];
E=[E1 E2];
de_v=zeros(1,length(v));
for k=1:length(v)
    de_v(k)=dehillspline3(v(k),v,E);
end
plot(v,de_v,'-o')
xlabel('v(AgNO_3)/mL');ylabel('dE/dv')
```

得到的 0.1mol/L 的 $AgNO_3$ 溶液滴定 NaCl 溶液的 dE/dv-v 曲线见图 5-5：

从图 5-5 看出，dE/dv-v 曲线确实在［5，25.5］之间存在一最高点，但这个最高究竟在哪里，目前还不能肯定。我们使用求函数最大值的 gax 函数确定最高点的位置。也就是本次电位滴定终点值。gax 函数是基于遗传算法求函数最大值的函数，我们在上一章曾讲到，它的基本使用格式如下：

[x_max,y_max,n_gen]=gax(f_name,bounds)

gax 函数输入和输出参数的意义如下：

f_name：M 文件名，定义要求解的函数，是一字符串常数

bounds：2 列矩阵，第一列和第二列分别存储矩形闭区域的各维度的下界和上界

x_max：最优点

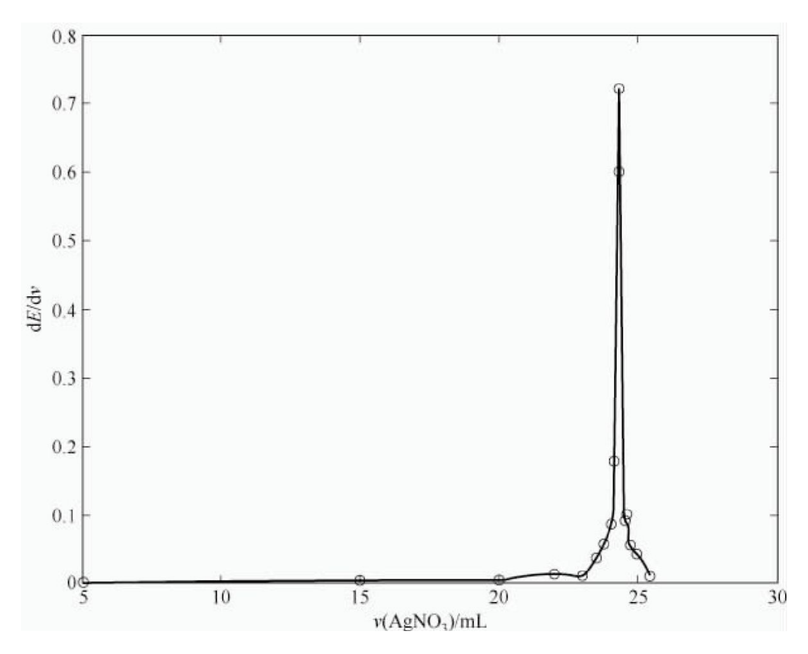


图 5-5　dE/dv-v 曲线

y _max：最大值

n _gen：繁殖代数

根据 gax 函数的使用方法，我们必须首先定义一个函数，该函数的作用是返回任意滴定剂体积时的 dE/dv 值，定义如下：

```
function dE_v=f(vx)
    v=[5 15 20 22 23 23.5 23.8 24 24.1 24.2 24.3 24.4 24.5 24.6 24.7 25 25.5];
    E1=[0.062 0.085 0.107 0.123 0.138 0.146];
    E2=[0.161 0.174 0.183 0.194 0.233 0.316 0.340 0.351 0.358 0.373 0.385];
    E=[E1 E2];
    dE_v=dehillspline3(vx,v,E);
```

在完成求任意滴定剂体积时的 dE/dv 值的函数后，我们就可以使用 gax 函数求 dE/dv-v 曲线的最大值点，注意这个最大值点在［5，25.5］之间，具体的 MATLAB 代码如下：

```
[v_max,dE_dv_max,n_gen]=gax('f',[ 5 25.5])
```

结果是：

```
v _max=24.3440
dE _dv _max=0.9182
n _gen=500
```

因此，0.1mol/L 的 $AgNO_3$ 溶液滴定该 NaCl 溶液达到滴定终点时，共消耗 0.1mol/L 的 $AgNO_3$ 溶液 24.34mL。

5.3 最小二乘法

若 y 是关于自变量 $X=(x_1\ x_2\ x_3\cdots x_n)$ 和参数 $B=(b_1\ b_2\ b_3\cdots b_n)$ 的形式已知的函数：

$$y=f(X,B)$$

今给出（X，y）的 n 对观测值：

$$(X_k,y_k)\qquad(k=1,2\cdots n)$$

要求确定参数 B 使得下面的函数为最小。

$$Q=\sum_{k=1}^{n}[y_k-f(x_k,B)]^2$$

这就是曲线拟合经验公式中参数的最小二乘估计问题。

自然，根据 $f(X,B)$ 是否是线性函数，最小二乘法可以分为线性最小二乘法和非线性最小二乘法。我们下面分别讲述线性最小二乘法和非线性最小二乘法。

5.3.1 线性最小二乘法

用 MATLAB 实现线性最小二乘法比较简单，其实质就是解一个线性矛盾方程组，我们在前面求解线性方程组的章节，已经讲述过如何求解线性矛盾方程组，因此，我们这里举两个例子，说明线性最小二乘法。

5.3.1.1 N_2(g)在 $ZrSO_4$(s)上的吸附

在液氮温度时，N_2(g)在 $ZrSO_4$(s)上的吸附符合 BET 公式。今取 17.53g 样品进行吸附测定，N_2(g)在不同平衡压力下的被吸附体积如表所示（所有吸附体积都已换算成标准状况），已知饱和压力 $p_s=101.325$kPa，吸附数据见表 5-8。试计算：①形成单分子层所需要的 N_2(g)体积；②每克样品的表面积，已知每个 N_2(g)分子的截面积为 0.162nm^2。

表 5-8　N_2(g)在 $ZrSO_4$(s)上的吸附数据

p/kPa	1.39	2.77	10.13	14.93	21.01	25.37	34.13	52.16	62.82
$V/(10^{-3}\text{dm}^3)$	8.16	8.96	11.04	12.16	13.09	13.73	15.10	18.02	20.32

根据 BET 吸附公式：

$$\frac{p}{V(p_s-p)}=\frac{1}{V_mC}+\frac{C-1}{V_mC}\frac{p}{p_s}$$

以 $\frac{p}{V(p_s-p)}$ 对 $\frac{p}{p_s}$ 作图，应该得到一直线，直线的斜率是 $\frac{C-1}{V_mC}$，直线的截距是

$\frac{1}{V_m C}$，由此可以得到 $V_m=\frac{1}{截距+斜率}$，V_m代表在固体表面上铺满单层分子时所需要的气体体积，从 V_m可求得形成单分子层所需要的 N_2(g)体积。

吸附剂的总表面积 $S=A_m Ln$。其中，A_m是一个吸附质分子的横截面积，L 是阿佛伽德罗常数，n 是吸附质的物质的量，由此可求出每克样品的表面积。

我们共有 9 组 N_2(g)在不同平衡压力下的被吸附体积的数据，自然也有 9 组 $\left[\frac{p}{V(p_s-p)}\frac{p}{p_s}\right]$数据，我们设 $y=\frac{p}{V(p_s-p)}$，$x=\frac{p}{p_s}$，因为 y 和 x 是直线关系，所以我们设：

$$y=kx+b$$

解决问题①、②的关键是求出 k 和 b，即直线的斜率和截距。

由 9 组 $[y\ x]$ 数据，我们能够得到 9 个关于 k、b 的线性方程，这实际上是关于 k、b 的矛盾线性方程组，正如我们以前提到的，MATLAB 按照最小二乘原理求解矛盾线性方程组，即求 k、b，使得下式的 Q 为最小。

$$Q=\sum_{m=1}^{9}[y_m-(kx_m+b)]^2$$

需要注意的是，对于该线性矛盾方程组：

$$y_m=kx_m+b*1 \qquad (m=1,2\cdots9)$$

它的系数矩阵是： 它的右端向量是：

x_1	1	y_1
x_2	1	y_2
⋮	⋮	⋮
x_9	1	y_9

它的未知量是 k 和 b，请读者务必注意。

实现线性最小二乘法，解决上述问题的 MATLAB 代码如下：

```
p=[1.39 2.77 10.13 14.93 21.01 25.37 34.13 52.16 62.82];
V=[8.16 8.96 11.04 12.16 13.09 13.73 15.10 18.02 20.32];
ps=101.325;
y=p./(V.*(ps-p));
x=p/ps;
a=[x' ones(length(x),1)];
b=y';
c=a\b;                    %解矛盾线性方程组
k=c(1);b=c(2);
Vm=1/(k+b)*1e-3       %形成单分子层所需要的N2(g)体积(升)
```

```
Sm=0.162 *(1e-9)^2 *6.02e23 *Vm/22.4/17.53    %每克样品的表面积(平方米/克)
tx=0:0.1:1;
ty=k *tx+b;
plot(x,y,'o',tx,ty)
xlabel('p/ps')
ylabel('p/(V *(ps-p))')
```

运行结果如下：

Vm=0.0082

Sm=2.0485

使用最小二乘法得到的直线与数据点的比较见图 5-6：

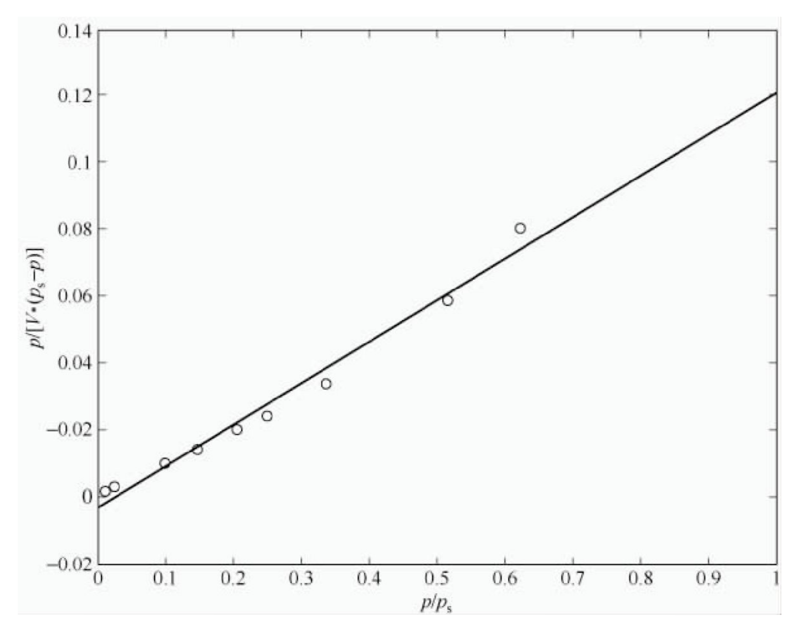


图 5-6　最小二乘法得到的直线与数据点的比较

因此，形成单分子层所需要的 $N_2(g)$ 体积是 $0.0082dm^3$，每克样品的表面积是 $2.0485m^2/g$。图 5-6 上的圆圈“o”代表原始数据点$\left[\frac{p}{V(p_s-p)}\ \frac{p}{p_s}\right]$，图中直线就是通过最小二乘法得到的直线，从图中可以看出，原始数据点大致均匀的分布在最小二乘直线直线两侧，最小二乘直线从所有数据点中间穿过，最小二乘直线是拟合所有具有线性关系的数据点的最佳直线。

5.3.1.2　多相催化反应动力学

某多相催化反应 $C_2H_{6(g)}+H_{2(g)}\xrightarrow{Ni/SiO_2}2CH_{4(g)}$，在 464K 时，测得数据见表 5-9。

表 5-9 多相催化反应动力学数据

p_{H_2}/kPa	10	20	40	20
$p_{C_2H_6}$/kPa	3.0	3.0	3.0	1.0
r/r_0	3.10	1.00	0.20	0.29

r 代表反应速率，r_0 是当 $p_{H_2}=20$kPa 和 $p_{C_2H_6}=3$kPa 时的反应速率。若反应速率公式可表示为 $r=kp_{H_2}^{\alpha}p_{C_2H_6}^{\beta}$，根据以上数据求 α 和 β 的值。

根据题意：

$$r=kp_{H_2}^{\alpha}p_{C_2H_6}^{\beta} \tag{1}$$

$$r_0=k20^{\alpha}3^{\beta} \tag{2}$$

式(2) 除以式(1)，得：

$$\frac{r}{r_0}=\left(\frac{p_{H_2}}{20}\right)^{\alpha}\left(\frac{p_{C_2H_6}}{3}\right)^{\beta} \tag{3}$$

我们有 4 组 [p_{H_2} $p_{C_2H_6}$ r/r_0] 数据，要求 α 和 β 的值，这是一个非线性最小二乘问题，即确定 α 和 β 的值，使得：

$$\sum_{k=1}^{4}\left[\frac{r_k}{r_0}-\left(\frac{p_{H_2,k}}{20}\right)^{\alpha}\left(\frac{p_{C_2H_6,k}}{3}\right)^{\beta}\right]^2$$

为最小。一般来讲，求解非线性最小二乘问题是有一定难度的，我们下面将要讲到。对于式(3)，我们可以在式(3) 两边取对数，使之转化为线性最小二乘问题，即：

$$Ln\left(\frac{r}{r_0}\right)=\alpha Ln\left(\frac{p_{H_2}}{20}\right)+\beta Ln\left(\frac{p_{C_2H_6}}{3}\right) \tag{4}$$

对于上述线性最小二乘问题，需要注意的是：

其系数矩阵是：　　　　　　其右端向量是：

$$\begin{matrix} Ln\left(\frac{p_{H_2}}{20}\right)_1 & Ln\left(\frac{p_{C_2H_6}}{3}\right)_1 & & Ln\left(\frac{r}{r_0}\right)_1 \\ Ln\left(\frac{p_{H_2}}{20}\right)_2 & Ln\left(\frac{p_{C_2H_6}}{3}\right)_2 & & Ln\left(\frac{r}{r_0}\right)_2 \\ \vdots & \vdots & & \vdots \\ Ln\left(\frac{p_{H_2}}{20}\right)_4 & Ln\left(\frac{p_{C_2H_6}}{3}\right)_4 & & Ln\left(\frac{r}{r_0}\right)_4 \end{matrix}$$

实现线性最小二乘法，解决上述问题的 MATLAB 代码如下：

```
pH2=[10 20 40 20 ];
PC2H6=[3.0 3.0 3.0 1.0];
r_r0=[3.10 1.00 0.20 0.29];
b=log(r_r0');
a=[log(pH2'/20) log(PC2H6'/3)];
x=a\b;
```

```
alpha=x(1)
beta=x(2)
```

运行结果是：

alpha=－1.9771

beta=1.1268

因此，α 和 β 的值分别近似为－2 和 1。

5.3.2　非线性最小二乘问题

如果 $f(X,B)$是非线性函数，则确定参数 B 使得：

$$Q=\sum_{k=1}^{n}[y_k-f(x_k,B)]^2$$

为最小的问题是非线性最小二乘问题。

非线性最小二乘问题实际是具有平方和形式的函数求最小值点的问题，可以应用我们以前讲述的牛顿-麦夸脱方法求解，自然，使用牛顿-麦夸脱方法要给出初值，这个初值由遗传算法产生。此外需要注意的是对于最小二乘法，定义函数向量时，未知量是参数 B。

5.3.2.1　SO_2 氧化动力学

制造硫酸的一步关键反应是：

$$SO_2+1/2O_2\longrightarrow SO_3$$

该反应用到固体催化剂，经过研究，建议的动力学速率方程的形式是：

$$r=\frac{k_1[SO_2][O_2]}{1+k_2[SO_3]}$$

有如下的测试数据，见表 5-10。

表 5-10　SO_2 氧化动力学数据

$r/mol\cdot m^{-3}\cdot s^{-1}$	9.70	12.13	14.29	16.30	18.23	20.11	21.96	23.79	25.60
$[SO_2]/ mol\cdot m^{-3}$	1.00	1.50	2.00	2.50	3.00	3.50	4.00	4.50	5.00
$[O_2]/ mol\cdot m^{-3}$	4.00	4.50	5.00	5.50	6.00	6.50	7.00	7.50	8.00
$[SO_3]/ mol\cdot m^{-3}$	1.00	1.50	2.00	2.50	3.00	3.50	4.00	4.50	5.00

试根据上表数据求出动力学方程中的常数 k_1 和 k_2。

这显然是一个非线性最小二乘问题，根据问题给出的函数形式，是可以把该函数形式变形，转化为线性最小二乘法，但我们这里直接采用求解非线性最小二乘问题的方法解决。

首先定义函数向量如下：

```
function y=f(k)
%k:二元向量，其每个向量分别代表速率方程中的参数 k1 和 k2
r=[9.70 12.13 14.29 16.30 18.23 20.11 21.96 23.79 25.60];  %反应速率数据
C_SO2=1:0.5:5;                                              %二氧化硫浓度数据
C_O2=4:0.5:8;                                               %氧气浓度数据
C_SO3=1:0.5:5;                                              %三氧化硫浓度数据
y=zeros(1,length(r));
y=r-k(1)*C_SO2.*C_O2./(1+k(2)*C_SO3);                     %定义函数向量
```

请读者注意，对于非线性最小二乘问题，函数向量的各分量的形式不是 $f(X)$，而是 $y_k-f(x_k,B)$。

在使用牛顿-麦夸脱方法前，自然先需要用遗传算法产生用于牛顿-麦夸脱方法的迭代初值，实现遗传算法的gax函数要求给出自变量的范围，根据动力学方程的特点，设两个参数介于［－20 20］。

由于gax是求函数最大值，因此求具有平方和形式的函数 F 的最小值点，相当于求函数 $-F$ 的最大值点，这点请读者务必注意。

解决上述问题的MATLAB代码如下：

```
h=@(b)-dot(f(b),f(b));          %定义函数-F
[x_max,y_max,n_gen]=gax(h,[-20 20; -20 20]); %利用遗传算法求迭代初值
[x_min,y_min]=marquet_F2('f',x_max)      %利用麦夸脱方法解非线性最小二乘
```

解得：

```
x_min=
    8.0044
    2.3013
y_min=6.3597e-005
```

所以，反应速率方程的动力学参数 $k_1=8.0044$，$k_2=2.3013$。

5.3.2.2 酒精在人体血液中分解的动力学

体重约70kg的某人在短时间内喝下2瓶啤酒后，隔一定时间测量他的血液中酒精含量（毫克/百毫升），得到数据见表5-11。

表5-11 酒精在人体血液中分解的动力学数据

时间/小时	0.25	0.5	0.75	1	1.5	2	2.5	3	3.5	4	4.5	5
酒精含量	30	68	75	82	82	77	68	68	58	51	50	41
时间/小时	6	7	8	9	10	11	12	13	14	15	16	
酒精含量	38	35	28	25	18	15	12	10	7	7	4	

另外根据酒精在人体血液分解的动力学可知，血液中酒精浓度与时间的关系可表示为：

$$c(t)=p(e^{-qt}-e^{-rt})$$

试根据上表数据求出参数 p、q 和 r 并 $c(t)$随时间变化的曲线图。

这显然是一个非线性最小二乘问题，根据问题给出的函数形式，无法把该函数形式线性化转化为线性最小二乘法，因此，必须使用求解非线性最小二乘问题的方法解决。

首先定义函数向量如下：

```
function y=f(b)
t=[0.25 0.5 0.75 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0];          %时间(小时)
c=[30.0 68.0 75.0 82.0 82.0 77.0 68.0 68.0 58.0 51.0 50.0 41.0 38.0 35.0 28.0 25.0 18.0 15.0 12.00 10.0 7.0 7.0 4.0];     %血液中酒精含量(毫克/百毫升)
y=zeros(1,length(t));
for k=1:length(t)
y(k)=c(k)-(b(1)*(exp(-b(2)*t(k))-exp(-b(3)*t(k))));  %定义函数向量
end
```

请读者注意，对于非线性最小二乘问题，函数向量的各分量的形式不是 $f(X)$，而是 $y_k-f(x_k,B)$。

在使用牛顿-麦夸脱方法前，自然先需要用遗传算法产生用于牛顿-麦夸脱方法的迭代初值，实现遗传算法的 gax 函数要求给出自变量的范围，根据 $c(\mathrm{t})$函数形式的特点，参数 p 是一普通系数，因此认为 p 介于 [−1000 1000]，q、r 位于指数项上，因此认为 p、q 介于 [−20 20]。

由于 gax 是求函数最大值，因此求具有平方和形式的函数 F 的最小值点，相当于求函数 $-F$ 的最大值点，这点请读者务必注意。

解决上述问题的 MATLAB 代码如下：

```
t=[0.25 0.5 0.75 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0];
c=[30.0 68.0 75.0 82.0 82.0 77.0 68.0 68.0 58.0 51.0 50.0 41.0 38.0 35.0 28.0 25.0 18.0 15.0 12.00 10.0 7.0 7.0 4.0];
h=@(b)-dot(f(b),f(b));                    %定义函数-F
[x_max,y_max,n_gen]=gax(h,[-1000 1000;-10 10;-10 10]);
                                          %利用遗传算法求迭代初值
```

```
[x_min,y_min]=marquet_F2('f',x_max)        %利用麦夸脱方法解非线性最小二乘
h2=@(x) x_min (1). *(exp(-x_min (2). *x)-exp(-x_min (3). *x));
tx=0:0.1:17;
yt=h2(tx);
plot(t,c,'o', tx, yt)
xlabel('时间/h')
ylabel('血液中的酒精浓度/mg/100mL')
```

结果是：

```
          114.4325
x _min=  0.1855
           2.0079
y _min=225.3417
```

因此 p=114.4325，q=0.1855，r=2.0079

我们也可使用 fminsearch 局部优化函数求出参数 p、q 和 r 的值：

代码如下：

```
t=[0.25 0.5 0.75 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 6.0 7.0 8.0 9.0 10.0
11.0 12.0 13.0 14.0 15.0 16.0];
c=[30.0 68.0 75.0 82.0 82.0 77.0 68.0 68.0 58.0 51.0 50.0 41.0 38.0 35.0
28.0 25.0 18.0 15.0 12.00 10.0 7.0 7.0 4.0];
h=@(b)-dot(f(b),f(b));
[x_max,y_max,n_gen]=gax(h,[-1000 1000; -10 10; -10 10]); %利用
遗传算法求迭代初值
h=@(b)dot(f(b),f(b));
xmin=fminsearch(h,x_max)
ymin=h(xmin)
```

结果是：

```
        114.4325
xmin=  0.1855
         2.0079
ymin=225.3417
```

在这里，我们使用了遗传算法+fminsearch 函数解该最小非线性二乘问题，得到的结果和遗传算法+麦夸脱方法得到的结果是相同的。

利用求得的参数 p、q 和 r 的值，做最小二乘曲线，见图 5-7。

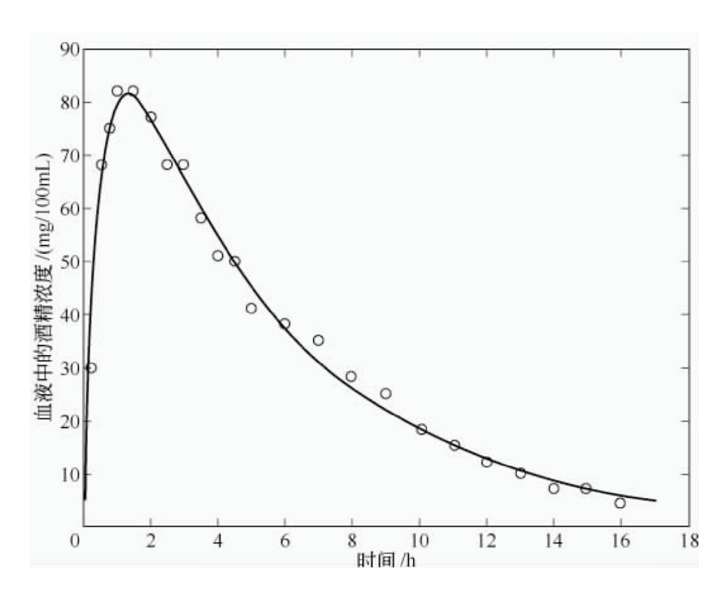


图 5-7　酒精在人体血液中分解的动力学曲线

5.4　基于统计学的数据处理方法

统计学是进行数据处理的强有力手段，利用统计学，能够对获得的实验数据进行描述性分析，并进一步进行统计推断，获取新的信息。MATLAB 有专门的数理统计工具箱（Statistics Toolbox）对统计学问题进行处理。

5.4.1　数据点的平均值、标准差与置信区间

数据的平均值反映了数据的，数据的标准差和变异系数反映了数据点的波动程度或离散程度。假设 x 是向量，x 的任一分量代表一组测量数据中某一数据点，则在 MATLAB 中，求数据点的平均值和标准差的函数分别 mean(x)和 std(x)，数据的变异系数的是 std(x)/mean(x)。

因为数据的平均值毕竟不是数据的真实值，在很多情况下，还需要根据数据的平均值和标准差来估计真实值所在的范围或区间。这个区间称为置信区间。在确定置信区间前，需要指定置信度，置信度是个概率值，它表明真实值有多大可能性落入置信区间内。一般来说，置信度越大，得到的置信区间越长，置信度越低，置信区间越短。我们当然希望置信区间短些好，但较短的置信区间对应较低的置信度，因此，一般置信度定为 95%。

求真值的置信区间，涉及比较多的统计学知识，请读者参考有关统计学教材，我们这里直接给出求真实值置信区间的函数 confidence_interval，如下：

```
function a_b=confidence_interval(x,p)
        %求在一定置信度下的置信区间
        %x:向量，每个分量代表一个测量数据点
        %p:置信度
        %a_b:二元向量，第一分量代表区间上限，第二分量代表区间下限
miu=mean(x);
sigma=std(x);
n=length(x);
lbd_negative=tinv((1-p)/2,n-1);
a_b=[miu+ lbd_negative *sigma/sqrt(n),miu- lbd_negative *sigma/sqrt(n)];
```

confidence_interval函数调用了MATLAB的统计学工具箱的tinv函数，该函数称为t分布的逆累计分布函数，该函数的具体使用方法请参考MATLAB的帮助文档和有关统计学教材。

5.4.1.1 铁矿中铁的含量平均值、标准差

分析铁矿中铁的含量，得到如下数据：37.45%，37.20%，37.50%，37.30%，37.25%，计算该组数据的平均值、标准差和变异系数。

具体的MATLAB代码如下：

```
x=[37.45,37.20,37.50,37.30,37.25]/100;  %数据点：铁矿石中铁的含量
miu=mean(x)                              %求平均值
sigma=std(x)                             %求标准差
CV=sigma/miu                             %求变异系数
```

结果是：
miu=0.3734
sigma=0.0013
CV=0.0035

由于变异系数是标准差除以平均值，因此可看做相对标准差，能够更好地反映数据的波动程度或离散程度。

5.4.1.2 SiO_2百分含量的置信区间

测定SiO_2的百分含量，得到如下数据：28.63，28.59，28.51，28.48，28.52，28.63。求置信度分别为90%和95%的置信区间。

求解上述问题的MATLAB代码如下：

```
x=[28.63 28.59 28.51 28.48 28.52 28.63];
a_b_90=confidence_interval(x,0.9)
a_b_95=confidence_interval(x,0.95)
```

结果是：

a _b _90＝28.5064　28.6136

a _b _95＝28.4917　28.6283

因此，真实值有 90％的可能性落入区间［28.5064 28.6136］，而真实值落入区间［28.4917 28.6283］的可能性是 95％。

5.4.2　假设检验

通俗地说，假设检验就是从样本值出发去判断一个“看法”是否成立，进行假设检验的方法是采用某种带有概率性质的反证法。假设检验的内容很丰富，请读者参考有关概率统计教材，我们在这里举两个例子，说明如何利用 MATLAB 进行假设检验。

5.4.2.1　判断测温仪器的系统误差

用某仪器测量温度，得到 5 个数据：1250，1265，1245，1260，1275。根据别的精确方法得到温度的真值是 1277，现在问题是：这台仪器测量温度有无系统误差？

从直观上看，这台仪器测温存在系统误差，如何给出合理而论据充分的判断呢？

为了解决这个问题，我们可以应用具有概率性质的反证法，即假设该仪器不存在系统误差，其测量温度真值就是 1277，在这种假设下，我们看一看得到 5 个温度数据（［1250，1265，1245，1260，1275］）的概率是多少？

我们用单样本 t 检验来计算这个概率，在 MATLAB 中，我们使用 ttest 函数执行单样本 t 检验。

ttest 的基本使用格式如下：

［h，p］＝ttest（x，m）

x 是向量，其每一分量代表一个测量数据点，m 是该组数据的假设的真实值。对于 ttest 函数的输出，我们只需知道 p 代表当该组数据的假设真实值取 m 时，得到一组测量数据 x 时的概率，一般来说，如果该概率 p 小于 0.05，就认为该组数据的真实值是 m 这一假设是不正确的，应该否定。如果 p 大于 0.05，就认为不能否定该组数据的假设真实值取 m 是错误的。

解决上述问题的 MATLAB 代码如下：

```
x=[1250 1265 1245 1260 1275];
m=1277;
[h,p]=ttest(x,m);
p
```

结果是：

p＝0.0280

上述单样本 t 检验给出 $p=0.0280$，其意义是：当假设该仪器不存在系统误差，其测量温度真值就是 1277，在这种假设下，该仪器测量得到的 5 个温度数据（[1250，1265，1245，1260，1275]）的概率是 $0.0280<0.05$，这是一个概率很小的事件，一般来说，小概率事件在一次观察中可以认为基本不会发生，但小概率事件恰恰在一次观察中发生了，这与我们的认知原则（小概率事件在一次观察中可以认为基本不会发生）是矛盾的，所以，基于此，我们否定假设，从而判断该仪器存在系统误差。

5.4.2.2 水处理工艺比较

某工厂采用新法处理废水，对处理后的水测量所含某种有毒物质的浓度，得到 10 个数据（单位：毫克/升）：22，14，17，13，21，16，15，16，19，18。

而以往老法处理废水后，该种有毒物质的平均浓度是 19，问新法是否比老法效果好？

为了解决这个问题，我们仍然可以应用具有概率性质的反证法，即假设新法没有老法效果好，即新法处理废水后，有毒物质的浓度 $\geqslant 19$，如何看一看在这种假设下，能以多大概率得到新法处理废水后，得到上述的 10 个数据。

这个概率同样使用 ttest 函数完成，具体使用格式如下：

[h,p]=ttest(x,m,alpha,'right')

x 是向量，其每一分量代表一个测量数据点，m 是要与 x 的真实值相比较的数值，alpha 代表小事件的概率，一般认为 alpha=0.05 时，该事件就是小概率事件，字符'right'代表对假设 x 的真实值 $\leqslant m$ 的计算概率 p。自然，如果 $p<$alpha，就否定假设。

进行新老工艺比较的 MATLAB 代码如下：

```
x=[22,14,17,13,21,16,15,16,19,18];
m=19;
[h,p]=ttest(x,m,0.05,'right');
p
```

结果是：

p=0.9650

由于 p>alpha=0.05，因此我们不能否定假设，即新法的处理效果比老法好。

以上判断系统误差、比较实验工艺，均属于显著性检验或假设检验的内容，即比较数据之间是否存在显著性差异，如果存在显著性差异，可以进一步比较数据真实值或方差的大小，有关内容请读者参考有关数理统计教材和 MATLAB 的帮助文档。

参考文献

[1] 阮复昌，关建郁．化工过程的数学模型及计算机模拟．广州：华南理工大学出版社，2001．

[2] 张建侯，许锡恩．化工过程分析与计算机模拟．北京：化学工业出版社，1989．

[3] 冯康．数值计算方法．北京：国防工业出版社，1978．

[4] 甄西丰．实用数值计算方法．北京：清华大学出版社，2006．

[5] William H. Press，Saul A. Teukolsky，William T etal．Numerical Recipes 3rd Edition：The Art of Scientific Computing．Cambridge：Cambridge Press，1985．

[6] 谭浩强，田淑清．FORTRAN 语言程序设计，北京：高等教育出版社，1986．

[7] 刘柄文．quick basic 程序设计（修订版）．北京：电子工业出版社，1998．

[8] 中村小一郎［美］著．科学计算引论-基于 MATLAB 的数值分析（第二版）．梁恒译．北京：电子工业出版社，2002．

[9] Mathews，J. H.［美］著．数值方法（MATLAB 版）（第四版）．周璐译．北京：电子工业出版社，2005．

[10] Cleve Moler．Numerical Computing with MATLAB．New York：the Society for Industrial and Applied Mathematics，2004．

[11] http：//www. mathworks. com/

[12] H. 斯科特．福格勒［美］著．化学反应工程（原著第 3 版）．李术元译．北京：化学工业出版社，2006．

[13] 朱开宏，袁渭康．化学反应工程分析．北京：高等教育出版社，2002．

[14] E. Bruce Nauman［美］著．化学反应器的设计、优化和放大．朱开宏，李伟，张元兴译．北京：中国石化出版社，2004．

[15] 华东化工学院分析化学教研组，成都科技大学分析化学教研组．分析化学（第三版）．北京：高等教育出版社，1989．

[16] 大连理工大学无机化学教研室．无机化学解题与思考．大连：大连理工大学出版社，1990．

[17] 傅献彩，沈文霞，姚天扬等．物理化学（第五版）．北京：高等教育出版社，2005．

[18] J. B. 登斯著［美］．化学中的数学方法．王知群译．北京：科学出版社，1981．

[19] E. L. 鲍尔［美］著．化学用数理统计手册．王铮，邓时俊译．北京：化学工业出版社，1983．

[20] 孙德敏．工程最优化方法及应用．合肥：中国科学技术大学出版社，1991．